Metode entropijskog kodiranja

Ožuška, Mario

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ODJEL ZA FIZIKU

MARIO OŽUŠKA

METODE ENTROPIJSKOG KODIRANJA

Diplomski rad

Osijek, 2016.

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ODJEL ZA FIZIKU

MARIO OŽUŠKA

METODE ENTROPIJSKOG KODIRANJA

Diplomski rad

predložen Odjelu za fiziku Sveučilišta J.J. Strossmayera u Osijeku
radi stjecanja zvanja profesora fizike i informatike

Osijek, 2016.

Ovaj diplomski rad izrađen je u Osijeku pod vodstvom izv.prof.dr.sc. Darka Dukića u sklopu Sveučilišnog diplomskog studija Fizike i informatike na Odjelu za fiziku Sveučilišta Josipa Jurja Strossmayera u Osijeku.

SADRŽAJ

Sveučilište J.J. Strossmayera u Osijeku **Diplomski rad**
Odjel za fiziku

METODE ENTROPIJSKOG KODIRANJA

MARIO OŽUŠKA

Sažetak

Kodiranje predstavlja transformaciju neke poruke ili općenito nekog objekta na način da se simboli (slova, brojevi, pikseli, ...) tog objekta zamjene simbolima neke druge abecede (npr. slova se zamjene brojevima). Metode entropijskog kodiranja vrše kodiranje bez gubitka, a temelje se izravno na teoriji informacije. Osim toga, neke od metoda postižu vrlo efikasan omjer kompresije što ih uz svojstvo kodiranja bez gubitka čini idealnim metodama kada se mora sačuvati originalan oblik sadržaja. U ovom radu izloženo je pet metoda entropijskog kodiranja: Shannon-Fanovo kodiranje, Huffmanovo kodiranje, aritmetičko kodiranje, metode rječnika i metode skraćivanja niza.

(50 stranice, 5 slika, 29 literaturnih navoda)
Rad je pohranjen u knjižnici Odjela za fiziku
Ključne riječi: teorija informacije / kodiranje / kompresija / metode entropijskog kodiranja
Mentor: izv.prof.dr.sc. Darko Dukić
Ocjenjivači: izv.prof.dr.sc. Branko Vuković, doc.dr.sc. Igor Lukačević
Rad prihvaćen: 5. listopada 2016.

J.J. Strossmayer University in Osijek **Master Thesis**
Department of Physics

METHODS OF ENTROPY CODING

MARIO OŽUŠKA

Abstract

Coding represents the transformation of some message or generally an object in the way that symbols (letters, numbers, pixels, ...) of that object being replaced with symbols of some other alphabet (for example, letters being replaced with numbers). Methods of entropy coding perform coding without loss, i.e. lossless compression. They are based directly on information theory. In addition, some of the methods achieve very efficient compression ratio which makes them ideal for coding when it is important to keep the original form of the content. In this thesis, five methods of entropy coding are presented: Shannon-Fano coding algorithm, Huffman coding algorithm, arithmetic coding algorithm, dictionary methods, and simple repetition suppression.

(50 pages, 5 figures, 29 references)
Thesis deposited in Department of Physics library
Keywords: information theory / coding / compression / methods of entropy coding
Supervisor: Darko Dukić, PhD, Associate Professor
Reviewers: Branko Vuković, PhD, Associate Professor, Igor Lukačević, PhD, Assistant Professor
Thesis accepted: October 5, 2016

1. UVOD

Zamislite da sjedite u svojoj omiljenoj fotelji i počinjete gledati film, koji ste daljinskim upravljačem pokrenuli s DVD playera. I kako film odmiče, pomislite da bi doživljaj gledanja možda bio potpuniji kada bi istovremeno slušali neku omiljenu glazbu pa stavljate slušalice i pokrećete MP3 player. Odjednom vam ugođaj prekine vibracija u lijevom džepu. Posegnete rukom u džep i izvadite mobitel, a na ekranu piše da imate novu poruku. Sa smiješkom je čitate jer vam najbolji prijatelj javlja kako mu je stigla plaća, a on je upravo sa s bankomata, pomoću kartice, podigao gotovinu te vas zove večeras na piće u obližnji kafić. Budući da imate vremena, vraćate se gledanju filma. U jednoj sceni uočite glumicu koja vam se učinila poznatom pa odlučite to provjeriti. Ponovno uzimate mobitel, povezujete se na internet te u pretraživaču upisujete ime glumice. Otvarate jednu od ponuđenih stranica i povećate fotografiju kako bi bili sigurni da niste pogriješili. Zadovoljni što ste bili u pravu, vraćate se gledanju filma, ali vam se više ne čini zanimljivim pa daljinskim upravljačem počnete prebacivati programe na TV-u. Budući da se bliži vrijeme kada ste dogovorili susret s prijateljem, spremate se i izlazite iz stana. Naravno, sa sobom nosite mobitel i nekoliko bankovnih kartica.

Iako većina ljudi o tome ne razmišlja, radnje koje su prethodno opisane temelje se na teoriji informacije, odnosno na korištenju različitih metoda kodiranja i dekodiranja. Kada je pomoću daljinskog upravljača poslan signal DVD playeru, on je prepoznat kao naredba za pokretanje filma. Film je na DVD-u spremljen u kodiranom obliku pa ga je potrebno dekodirati kako bi mogao biti prikazan na TV ekranu. Isti se proces odvija i prilikom puštanja glazbe s MP3 playera te kada komuniciramo preko mobitela. Digitalne fotografije na internetu također su kodirane pa je za njihovo gledanje potrebno imati odgovarajući alat, odnosno dekoder. Ni gotovinu s bankomata nećemo moći podići bez odgovarajuće kartice koju uređaj prvo mora prepoznati. No, zašto je uopće potrebno vršiti kodiranje? Razloga je više pa se s obzirom na namjenu primjenjuju različite vrste kodiranja. U slučaju bankovne kartice od ključne je važnosti sigurnost te će se zbog toga koristiti kriptografske metode zaštite. Kompresija se koristi kako bi se smanjio prostor potreban za pohranu digitalnih sadržaja te skratilo vrijeme njihovog prijenosa. Sa svrhom reduciranja pogreški prilikom prijenosa podataka primjenjuju se zaštitno kodiranje. Naravno, u

praksi su često istovremeno prisutni svi navedeni razlozi pa se stoga kombiniraju različite metode.

Metode kompresije se jednostavno mogu podijeliti u dvije skupine: metode entropijskog kodiranja, koje vrše kompresiju bez gubitka, i metode izvornog kodiranja, koje najčešće vrše kompresiju s gubicima. U ovom su radu analizirane metode entropijskog kodiranja. Pri tome su prvo objašnjeni osnovni pojmovi važni za razumijevanje ove problematike, poput entropije, prosječne duljine kodne riječi te prefiksnog i optimalnog koda. Zatim je prikazano pet metoda entropijskog kodiranja: Shannon-Fanovo kodiranje, Huffmanovo kodiranje, aritmetičko kodiranje, metode rječnika i metode skraćivanja niza.

2. OSNOVNI POJMOVI ENTROPIJSKOG KODIRANJA

Entropijsko kodiranje je bazirano na činjenici da svaki signal ima jedinstvenu informaciju, a prosječna duljina koda vezana je za entropiju izvora informacije, što je poznato kao Shannonov prvi teorem. Kodna riječ za svaki simbol ne mora biti stalne duljine, nego duljina može biti varijabilna ovisno o količini entropije.[1]

Osnovna ideja entropijskog kodiranja je predstaviti vjerojatnije kvantizacijske indekse s kraćim kodnim riječima, a one manje vjerojatne s duljim kodnim riječima da bi se postigla manja prosječna duljina kodne riječi. Na ovaj način skup kvantizacijskih indeksa može biti prikazan s manjim brojem bitova.[2]

Entropijsko kodiranje je proces bez gubitka ili reverzibilan proces koji postiže dodatnu kompresiju kodirajući elemente sintakse u konačnoj izlaznoj datoteci. Kodovi varijabilne duljine, poput Huffmanovog ili aritmetičkog koda, statistički su kodovi koji imaju široki raspon korištenja.[3]

2.1. Entropija

Srednji vlastiti sadržaj informacije $I(X)$ može se shvatiti kao onaj iznos informacije koji je u prosjeku potreban da bi se odredio bilo koji pojedinačni simbol ili vijest iz skupa simbola X koji se predaju komunikacijskom sustavu. Veličina $I(X)$ naziva se entropijom diskretne slučajne varijable X i označava s $H(X)$.[4] Tako je entropija uvedena kao neodređenost, a ukupan sadržaj informacije upravo ovisi o tome kolika je neodređenost promatranog skupa, odnosno definirana je entropijom. U fizici entropija predstavlja mjeru za neuređenost sustava. Slično, entropija u teoriji informacije označava kvantitativnu mjeru neizvjesnosti sustava, odnosno nedostatak informacija o njemu. Ovo je vrlo jednostavno objasniti na primjeru bacanja novčića. Dakle, ako

[1] Wu, H.R., Rao, K.R. (ur.): Digital Video Image Quality and Perceptual Coding. Boca Raton: CRC Press, 2006., str. 12.
[2] You, Y.: Audio Coding: Theory and Applications. New York: Springer, 2010., str. 143.
[3] Bing, B.: Next-Generation Video Coding and Streaming. Hoboken: Wiley, 2015., str. 47.
[4] Sinković, V.: Informacija, simbolika i semantika: načela i primjena teorije informacije. Zagreb: Školska knjiga, 1997., str. 32.

bacimo novčić, prije nego što on padne na zemlju postoje dvije mogućnosti krajnjeg ishoda, a to su da padne pismo ili da padne glava (ako nismo baš te sreće da nam se novčić zaustavi po strani bočno). Dok je novčić u zraku postoji neizvjesnost u pogledu ishoda koja je razriješena nakon njegovog pada kada je, prema teoriji informacije, primljen 1 bit. Ako uzmemo u obzir obje mogućnosti ishoda, svaka od njih može se predstaviti sa 0.5 bit/simbol. Dok ne padne novčić mi ne znamo ishod pa je entropija (neodređenost) 0.5 bit/simbol. Kada novčić padne i mi saznamo rezultat, entropija je 0 bit/simbol. Razlog je tome što nam je tada poznat ishod bacanja i više nema neodređenosti. Matematički se entropija opisuje na sljedeći način:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \; [bit/simbol]$$

Ako postoji jednaka vjerojatnost pojavljivanja glave i pisma, tada entropija iznosi 0.5 bit/simbol. Kada bi imali novčić s dvije glave ili dva pisma, entropije ne bi bilo, jer je ishod siguran (opet, ako nismo baš te sreće da nam novčić padne i zadrži se na bočnoj strani).

2.2. Kodiranje

Osnovna svrha kodiranja informacije koju generira određeni izvor je osigurati što brži i pouzdaniji prijenos informacije od izvora do primaoca. U koderu i dekoderu signala želi se postići da se porukama izvora pridruže što je moguće kraći nizovi kodnih simbola, a da istovremeno vjerojatnost točnog dekodiranja bude što je moguće veća.[5] Samo kodiranje je jednostavno definirati. Ono predstavlja dodjeljivanje nekoj poruci kodnih riječi koje su sastavljene od novih simbola neke druge abecede, različite od one korištene za izvornu poruku. Tako se zapravo poruka definirana u jednoj abecedi zamjenjuje simbolima iz druge abecede. Kodirati se mogu pojedinačni simboli ili poruka u cjelini. Samo kodiranje se provodi iz raznih razloga. Osnovni, koji ćemo susresti u entropijskom kodiranju, je kompresija, odnosno sažimanje.

[5] Pauše, Ž.: Uvod u teoriju informacije, treće izdanje. Zagreb: Školska knjiga, 2003., str. 74.

2.3. Kompresija (sažimanje)

Sama kompresija se zapravo oslanja na kodiranje. Sve metode kompresije na neki način kodiraju objekt koji sažimaju i pri tome mu samim tim postupkom smanjuju veličinu, odnosno broj bitova potrebnih za zapis tog objekta u računalnoj memoriji ili, jednostavno, prostora potrebnog za rad s tim objektom. Smanjenje broja bitova provodi se dodjeljivanjem kodnih riječi. Osnovna svojstva svake metode kompresije proizlaze iz sljedećih pitanja:[6]

1. Vrši li se kompresija s gubitkom ili bez gubitka?
2. Koji je njen omjer?

Sama kompresija se može vršiti bez gubitka ili s gubitkom. Ako je kompresija provedena s gubitkom to znači da se određene karakteristike izvornog nekodiranog objekta gube što opet znači da se svaki dekodirani simbol neće moći vratiti u svoje originalno izvorno stanje. Dakle, došlo je do gubitaka nekih simbola originalne poruke pri kompresiji. Kompresija bez gubitka s druge strane znači da se svaki simbol može vratiti u svoje originalno izvorno stanje, odnosno pri kodiranju kojim se kompresija provodi ništa se ne gubi. Omjer kompresije se vrlo jednostavno definira kao omjer veličine osnovnog objekta i veličine komprimiranog. Ako je omjer kompresije npr. 1:10, znači da je za svakih deset bitova originalnog objekta potreban 1 bit komprimiranog. Što je rezultat omjera manji, kompresija je veća. Sve metode entropijskog kodiranja provode kompresiju bez gubitaka.

2.4. Zašto entropijsko kodiranje?

Entropija ovisi o vjerojatnosti pojavljivanja određenog elementa u promatranom sadržaju. To npr. može biti vjerojatnost pojavljivanja nekog simbola na izvoru poruke. Pojedini simboli mogu se češće pojavljivati, dok se drugi mogu pojavljivati rjeđe. Ova vjerojatnost pojavljivanja će onda determinirati i entropiju događaja. Sve metode entropijskog kodiranja koriste upravo svojstvo izvora da ne emitira sve simbole s istom vjerojatnosti. Tako će simboli koji se češće

[6] Pandžić, I.S., Bažant, A., Ilić, Ž., Vrdoljak, Z., Kos, M., Sinković, V.: Uvod u teoriju informacije i kodiranje. Zagreb: Element d.o.o., 2007., str. 237.

pojavljuju biti kodirani s kraćim kodnim riječima, dok će oni koji su rjeđi biti kodirani s dužim kodnim riječima. Na taj se način postiže kompresija. Svim metodama entropijskog kodiranja su također zajednička sljedeća svojstva:[7]

1. Temelje se izravno na teoriji informacije.
2. Vrše kodiranje bez gubitka.
3. Omjer kompresije ovisi samo o statističkim svojstvima izvora informacije.
4. Poruka se promatra isključivo kao niz slučajnih vrijednosti.
5. Ne uzimaju se u obzir svojstva medija.

Dakle, u osnovi su sve metode entropijskog kodiranja potpuno određene svojstvom izvora, odnosno s vjerojatnošću pojavljivanja simbola na izvoru.

2.5. Prosječna duljina kodne riječi

Simbole originalne poruke definiramo kodnim riječima koje su također sastavljene od nekih drugih simbola te ovisno o tome koliko ih simbola čini imaju svoju duljinu. Zbroj duljina svih kodnih riječi pomnoženih s vjerojatnosti pojavljivanja simbola koje kodiraju predstavlja prosječnu duljinu kodne riječi te poruke. Prosječna duljina kodne riječi računa se po formuli:

$$L = \sum_{I=1}^{n} p(x_i) l(x_i) = \sum_{i=1}^{n} p_i \, l_i \; [bit/simbol],$$

gdje je l_i duljina pojedine kodne riječi, a p_i vjerojatnost pojavljivanja simbola kojemu je ta kodna riječ dodijeljena.

Jedan od glavnih problema teorije kodiranja je u tome da se istraže uvjeti i postupci za konstruiranje kodova koji omogućuju jednoznačno dekodiranje za koje će L imati minimalnu

[7] Pandžić, I.S., Bažant, A., Ilić, Ž., Vrdoljak, Z., Kos, M., Sinković, V.: Uvod u teoriju informacije i kodiranje. Zagreb: Element d.o.o., 2007., str. 239.

vrijednost.[8] Upravo u rješavanju ovog problema nalazi se temeljna smjernica za konstrukciju optimalnih kodova.

2.6. Prefiksni kodovi

Ako neku poruku kodiramo uporabom prefiksnog koda to znači da se niti jedan kod neće nastavljati na prošli niti će sljedeći kod biti nastavak trenutnog koda. Dakle, svi kodovi su jedinstveni i samostalni te je svaki simbol konačno i jednoznačno kodiran. To isto tako znači da je svaka poruka jednoznačno kodirana. Primjena prefiksnih kodova znatno olakšava proces kodiranja i dekodiranja pošto ne postoje „nizovi“ od više kodova, nego je svaki kod zaseban i može se zasebno dekodirati neovisno o ostalim kodovima u poruci. Sve metode entropijskog kodiranja primjenjuju upravo prefiksne kodove.

2.7. Optimalni kodovi

Kod je optimalan ako ima prosječnu duljinu kodne riječi veću od entropije za maksimalno 1 bit. Zapisano matematički to izgleda ovako:

$$H(X) \leq L < H(X) + 1.$$

Ova definicija proizlazi iz zahtjeva da prosječne duljine pojedinačnih kodnih riječi budu minimalne, a da istovremeno bude zadovoljena Kraftova nejednakost oblika:

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

[8] Pauše, Ž.: Uvod u teoriju informacije, treće izdanje. Zagreb: Školska knjiga, 2003., str. 81.

gdje su l_i duljine pojedinačnih kodnih riječi, a d je baza koda, odnosno broj simbola koji mogu biti u kodu. Svi kodovi koji zadovoljavaju ovu nejednakost su prefiksni kodovi. Ako uzmemo oba uvjeta za optimalne duljine kodnih riječi dobijem sljedeći izraz:

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

koji govori da prosječna duljina kodne riječi odgovara entropiji kada je kod optimalan. Ako uzmemo u obzir da kodovi ne moraju biti cjelobrojni, tražit ćemo da prosječna duljina kodne riječi bude unutar jednog bita entropije, što je upravo onaj prvi uvjet koji smo naveli i koji je ujedno konačno rješenje ovog postupka.

Svi kodovi entropijskog kodiranja su u teoriji optimalni kodovi, odnosno samo kodiranje, kako bi bilo efikasno, zahtjeva da budu. Ipak, sve metode u ovome ne uspijevaju uvijek (npr. Shannon-Fanova metoda). Efikasnost koda računa se na sljedeći način:

$$\epsilon = \frac{H(X)}{L} \leq 1$$

Prema tome, efikasnost koda predstavlja omjer entropije i prosječne duljine kodne riječi.

U nastavku će biti izloženo pet osnovnih metoda entropijskog kodiranja: Shannon-Fanovo kodiranje, Huffmanovo kodiranje, aritmetičko kodiranje, metode rječnika i metode skraćivanja niza.

3. SHANNON–FANOVO KODIRANJE

Huffmanovo kodiranje rezultira optimalnim kodom, ali računanje prosječne duljine kodne riječi može biti kompliciranо. Kodovi dobiveni primjenom Shannon–Fanovog kodiranja su blizu optimalnih, a njihovu prosječnu duljinu kodne riječi je lako izračunati.[9] *Prema tome, riječ je o žrtvi kako bi se dobila metoda jednostavnija za računanje.*

Claude Elwood Shannon i Robert Fano bili su pioniri teorije informacije, a Shannona se smatra i njezinim utemeljiteljem. Zajedno su došli do postupka kodiranja, koji je prema njima dobio naziv Shannon–Fanovo kodiranje. Ono predstavlja jednu od prvih metoda kodiranja, a koja je temeljena na teoriji informacije.

Claude Elwood Shannon

Rođen je 30. travnja 1916. godine u Petokeyu u saveznoj državi Michigan (SAD), a djetinjstvo je proveo u Gaylordu, također u Michiganu. Tamo je pohađao i srednju školu koju je završio 1932. godine. Otac mu je bio poslovni čovjek, a majka učiteljica jezika. Još kao dijete pokazivao je sklonost prema mehanici i elektronici te je sam izradio različite naprave poput broda kojim se upravljalo pomoću radiovalova ili telegrafa. Godine 1932. počeo je pohađati sveučilište u Michiganu. Nakon toga je nastavio studij na MIT-u gdje je studirao matematiku i inženjering. Tamo je magistrirao te naposljetku stekao i doktorat. Nakon završetka studija, 1941. godine, radio je u Bellovom laboratoriju kao matematičar istraživač na izradi telefona. Bavio se utvrđivanjem najboljeg načinu prijenosa signala. Primio je mnoge nagrade za svoj rad, a 1957. godine imenovan je profesorom komunikacijske znanosti i matematike. Također je bio autor mnogih izuma među kojima je i motorizirani pogo štap. Umro je 24. veljače 2001. godine u Medfordu u saveznoj državi Massachusetts (SAD).[10]

Video o Claude Elwood Shannonu u trajanju od 30 minuta: https://www.youtube.com/watch?v=z2Whj_nL-x8 (07.09.2016.)

[9] Jones, G.A., Jones, J.M.: Information and Coding Theory. London: Springer, 2000., str. 45.
[10] Wikipedia: Claude Shannon. URL: https://en.wikipedia.org/wiki/Claude_Shannon (07.09.2016.)

Danas se ova metoda ipak rijetko koristi, a jedna od razloga je što ne daje uvijek optimalan kod. Željena svojstva koda na kojima se zasniva ova metoda su sljedeća:[11]

1. Niti jedna kodna riječ ne smije biti prefiks druge kodne riječi
2. U kodiranoj poruci simboli 0 i 1 trebaju se pojavljivati jednaki broj puta

Prvo željeno svojstvo proizlazi iz definicije prefiksnog koda. Dakle, mi u svojoj poruci želimo jednoznačne i međusobno neovisne kodove koji se ne nastavljaju, niti si prethode, kako bi jednoznačno mogli kodirati željeni simbol, odnosno željenu poruku. Drugo željeno svojstvo zapravo označava maksimalnu entropiju, a ona je maksimalna ako je u razdiobi simbola svaki simbol jednako zastupljen. Ako su npr. u binarnom kodu nule i jedinice jednoliko raspoređene onda je entropija takvog koda maksimalna, što vodi kodiranju s otprilike 1 bit/simbol.

Robert Mario Fano

Rođen je 11. studenog 1917. godine u Torinu u Italiji. Pohađao je školu za inženjering u Torinu sve do odlaska u Ameriku 1939. godine. Doktorirao je 1947. godine na MIT-u gdje je i radio od 1941. godine, a postao profesor 1947. godine. Tijekom drugog svjetskog rata radio je na proučavanju mikrovalova i filtera, a od 1950. do 1953. radio je u Lincolnovom laboratoriju na tehnikama radara. Djelovao je i u raznim drugim područjima u kojima je publicirao mnoge radove. Bio je član je Nacionalne akademije za znanosti i Nacionalne akademije za inženjering, a 1976. godine primio je Shannonovu nagradu Društva teorije informacije IEEE. Edukacijska medalja istog društva dodjeljena mu je 1977. godine. Umro je u Naplesu, Florida (SAD) 13. srpnja 2016. godine.[12]

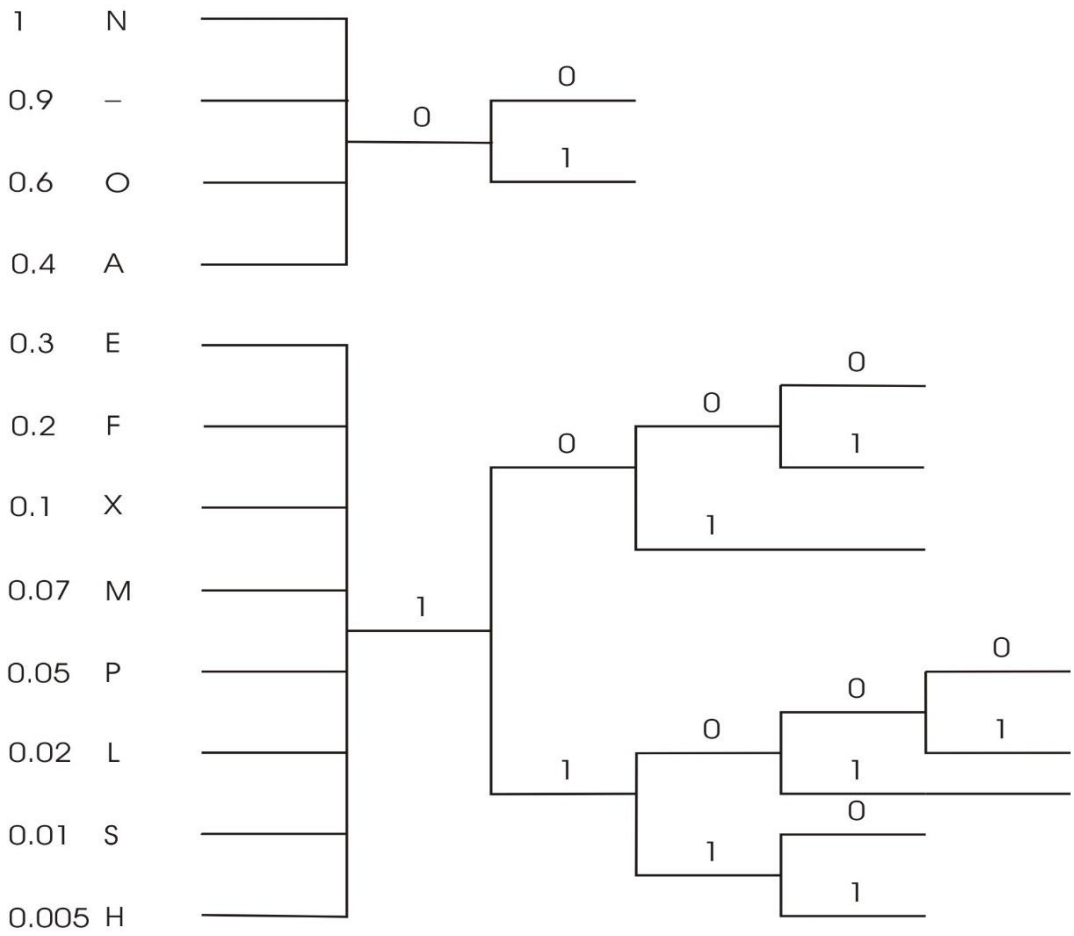
3.1. Postupak kodiranja

Sam postupak kodiranja mora ispuniti navedena željena svojstva koja su zapravo uvjeti za provođenje kodiranja. Kodiranje se odvija u sljedećim koracima:

[11] Pandžić, I.S., Bažant, A., Ilić, Ž., Vrdoljak, Z., Kos, M., Sinković, V.: Uvod u teoriju informacije i kodiranje. Zagreb: Element d.o.o., 2007., str. 253.
[12] Wikipedia: Robert Fano. URL: https://en.wikipedia.org/wiki/Robert_Fano (07.09.2016.)

1. Simboli se poredaju prema padajućoj vjerojatnosti pojavljivanja.
2. Simboli se podijele u dvije grupe tako da obje imaju podjednak zbroj vjerojatnosti.
3. Dodjeli se 0 prvoj grupi, a 1 drugoj grupi.
4. Ponavljaju se koraci 2 i 3 unutar svake grupe dok se grupa ne svede na jedan simbol.

Postupak Shannon–Fanovog kodiranje može se ilustrirati sljedećim binarnim stablom koje se zove Shannonovo stablo:



Slika 1. Shannonovo stablo

Simboli vidljivi s lijeve strane stabla poredani su prema padajućim vjerojatnostima pojavljivanja i podijeljeni prvo na dvije grupe. Zatim je svaka od tih grupa dijeljena dalje na dvije nove grupe i postupak je ponavljan dok na kraju nije ostao samo jedan simbol. Granama su pridruženi simboli 0 i 1.

3.2. Primjer Shannon–Fanovog kodiranja

Za primjer ćemo uzeti 6 simbola A, B, C, D, E i F s vjerojatnostima pojavljivanja p(A)=0.25, p(B)=0.25, p(C)=0.125, p(D)=0.125, p(E)=0.125 i p(F)=0.125. Prvo trebamo poredati ove simbole prema padajućim vjerojatnostima. Za to nije nužno koristiti binarno stabla. Ovdje će umjesto njega biti primijenjena tablica.

Simbol	Vjerojatnost	Korak 1	Korak 2	Korak 3	Kodna riječ	Duljina kodne riječi
A	0.25	0	0		00	2
B	0.25	0	1		01	2
E	0.125	1	0	0	100	3
F	0.125	1	0	1	101	3
C	0.125	1	1	0	110	3
D	0.125	1	1	1	111	3
Prosječna duljina kodne riječi:						2.5
Entropija:						2.5

Nakon što smo poredali simbole silazno prema vjerojatnostima, u prvom koraku smo podijelili sve simbole u dvije grupe, pri čemu je za obje grupe zbroj vjerojatnosti 0.5. Prvoj grupi smo, kako je vidljivo u tablici, dodijelili 0, a drugoj 1. Već ovaj korak osigurava da kodovi budu prefiksni, budući da npr. u ovom koraku simboli A i B nikako neće biti prefiksi simbola iz druge grupe pošto im je dodijeljen različit simbol. U drugom koraku smo sada ove dvije grupe podijelili na nove dvije grupe i ponovili postupak dodjele brojeva. Ovaj postupak je završen u trećem koraku pošto je ostao samo jedan simbol u svakoj grupi. Grupe koje su ostale na jednom simbolu u drugom koraku nisu dalje dijeljene, a dijeljenje je nastavljeno samo za one grupe s više simbola. I na kraju, očitavajući znamenke iz koraka, dobiva se binarni kod za svaki simbol. Prosječna duljina kodne riječi je 2.5, kao i entropija, pa zaključujemo da je kod optimalan i ima efikasnost 1.

4. HUFFMANOVO KODIRANJE

Huffmanov kod je široko korišten od kada je definiran od strane Huffmana 1952. godine. Međutim, ima nekoliko ograničenja. Prvo, zahtijeva da minimalno 1 bit predstavlja pojavljivanje svakog simbola. Zato ne možemo dizajnirati Huffmanov kod kojega bi dodijelili 0.5 bit/simbol. Stoga, ako je entropija izvora manja od 1 bit/simbol, Huffmanovo kodiranje nije efikasno. Čak i ako je entropija veća od 1, u većini slučajeva kod zahtijeva veći srednji broj bitova od entropije. Drugo, ne može se efikasno prilagoditi izvorima s promjenjivom statistikom. Iako su stvorene dinamične sheme Huffmanovog kodiranja kako bi riješile navedene probleme, njih je relativno teško promijeniti.[13]

Huffmanovo kodiranje je metoda kodiranja u kojoj se simbolima dodjeljuju kodne riječi različitih duljina, ovisno o njihovoj frekvenciji, odnosno vjerojatnostima pojavljivanja. Huffmanov kod je kod promjenjivih duljina zbog toga što se različitim simbolima dodjeljuju kodne riječi različitih duljina, odnosno ovaj kod dodjeljuje kraće kodove simbolima koji se češće pojavljuju, a duže kodove simbolima čije je pojavljivanje rjeđe. Shannon je dokazao da je prosječan broj bitova po uzorku u Huffmanovom kodu unutar 1 bita entropije:[14]

$$Entropy \leq R_{Huffman} \leq Entropy + 1$$

Samo kodiranje je efikasnije ako bitovi, odnosno simboli nisu jednoliko raspoređeni. Algoritam je smislio David Albert Huffman 1951. godine dok je bio student doktorskog studija, kada je dobio zadatak pronalaska najefikasnijeg načina kodiranja. Sam Huffman je tom prilikom zapravo smislio potpuno novi način kodiranja koji je bio efikasniji od svega što je u to vrijeme bilo poznato. Algoritam je publicirao 1952. godine.

[13] Mandal, M.Kr.: Multimedia Signals and Systems. Boston: Kluwer Academic Publishers, 2003., str. 133.
[14] Bosi, M., Goldberg, R.E.: Introduction to Digital Audio Coding and Standards. New York: Springer Sciance + Business Media, LLC, 2003., str. 41.

David Albert Huffman

Rođen je 9. kolovoza 1925 godine u Alliance, savezna država Ohio (SAD). Dao je jedan od temeljnih doprinosa matematičkom modelu računalnih znanosti dok je ona još bila u svojim povojima, a mnoge njegove tehnike, poput samog Huffmanovog kodiranja, aktualne su još i danas. Godine 1944. stekao je stupanj prvostupnika sveučilišta u Ohiou iz elektrotehnike. Nakon toga je služio u Američkoj mornarici kao časnik zadužen za održavanje na razaraču. Magistrirao je 1949. godine, također na sveučilištu u Ohiou, a doktorirao je 1953. godine na MIT-u. Doktorirao je iz područja elektrotehnike. Od 1953 radio je na MIT-u. Nakon 14 godina provedenih na MIT-u, 1967. godine nastavio je raditi na Sveučilištu Kalifornija u Santa Cruzu, kao jedan od osnivača odjela za informatiku. Bio je i voditelj navedenog odjela od 1970. do 1973. godine. Za svoj rad je primio više nagrada. U mirovinu je otišao 1994. godine, ali je i dalje držao tečajeve iz područja teorije informacije i analize signala. Umro je 7. listopada 1999. godine u bolnici u Santa Cruzu nakon desetomjesečne borbe s rakom.[15]

Novost koju je Huffman uveo u tom radu bila je konstrukcija binarnog stabla od dolje umjesto tada standardne konstrukcije koja je kretala od gore. Sam Huffmanov kod je zapravo binarno stablo koje se konstruira ovisno o frekvencijama pojavljivanja pojedinih simbola pri čemu se, posljedično, oni simboli koji se češće pojavljuju kodiraju kodnim riječima manje duljine, dok se oni koji se pojavljuju rjeđe kodiraju kodnim riječima veće duljine. Tako se postiže konstrukcija optimalnog koda na kojemu se i temelji cijelo Huffmanovo kodiranje. Zahvaljujući ovakvom načinu kodiranja Huffmanov kod postiže vrlo dobru kompresiju, u prosjeku oko 55% originalne veličine poruke. To je razlog zbog kojega Huffmanovo kodiranje ima vrlo raširenu primjenu u kompresiji slike ili videa (npr. HDTV koristi Huffmanovo kodiranje za kompresiju). Bitno je napomenuti da pri stvaranju koda, vjerojatnosti moraju biti unaprijed poznate inače samo kodiranje nije moguće. Osim toga, tablice koje Huffmanovo kodiranje koristi moraju biti poznate i koderu i dekoderu signala.

[15] Wikipedia: David A. Huffman. URL: https://en.wikipedia.org/wiki/David_A._Huffman (07.09.2016.)

Hufmanovo kodiranje se temelji na sljedeća dva teorema:[16]

1. U optimalnom kodu, simboli s većom vjerojatnosti pojavljivanja ne mogu imati dulje kodne riječi od onih s manjom vjerojatnosti pojavljivanja.
2. U optimalnom kodu, dva simbola s najmanjim vjerojatnostima pojavljivanja imaju kodne riječi iste duljine.

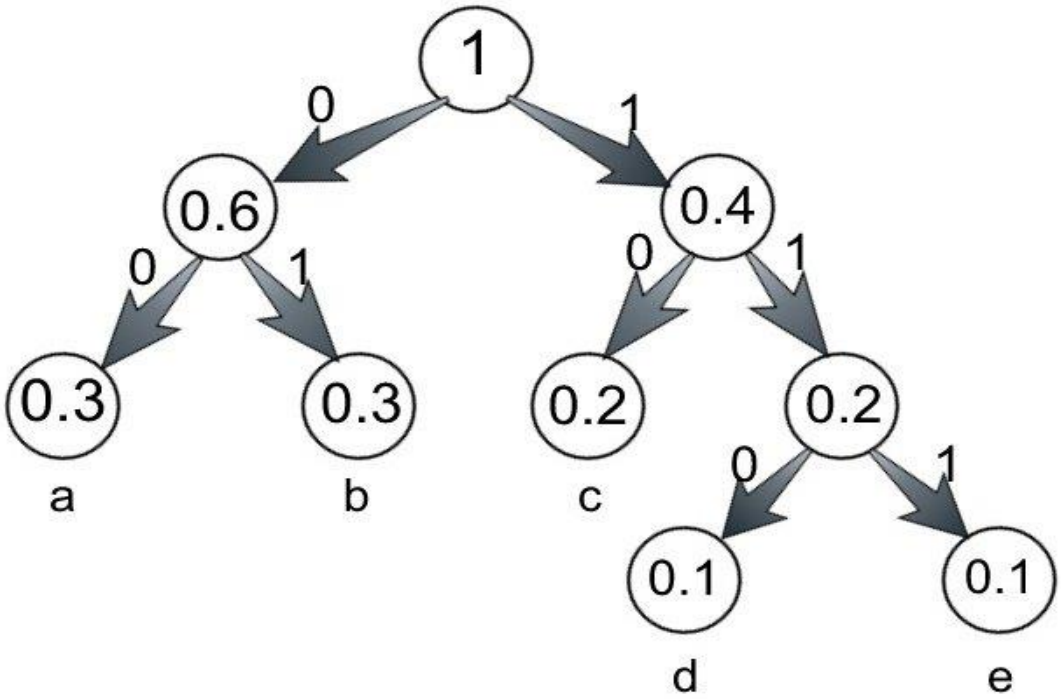
Kada ne bi bio istinit, prvi teorem bi kršio uvjete optimalnosti pa samim time ne bi mogao biti temelj Huffmanovog koda, za kojeg se pretpostavlja da mora biti optimalan. Drugi teorem također je jednostavno dokazati. Naime, ako su kodovi prefiksni, odnosno niti jedan kod ne može poslužiti kao prethodnik ili nastavak drugog koda, logično je zaključiti da u optimalnom kodu kodna riječ simbola s većom vjerojatnosti pojavljivanja ne može biti dulja od kodne riječi simbola s manjom vjerojatnosti pojavljivanja. Imajući to na umu, ako uzmemo dvije kodne riječi simbola s najmanjom vjerojatnosti pojavljivanja, one moraju imati dulje kodne riječi od bilo kojeg drugog simbola jer u suprotnom kod ne bi bio optimalan. Samim time one moraju biti jednake duljine jer ne mogu, odnosno ne moraju biti dulje, budući da svi ostali simboli imaju kraće kodne riječi.

4.1. Postupak kodiranja

Pri provođenju Huffmanovog kodiranja prvo uzmemo dva simbola s najmanjim vjerojatnostima pojavljivanja, za koje zbog istinitosti drugog teorema znamo da imaju kodne riječi jednake duljine, te ih označimo s 0 i 1. Ove vrijednosti pišemo na grane binarnog stabla koje konstruiramo, a ta dva simbola kombiniramo u jedan nad-simbol tako da im zbrajamo vjerojatnosti. Postupak nastavljamo tako da ovaj simbol postaje jedan čvor našeg binarnog stabla i njega također uspoređujemo s poznatim vjerojatnostima simbola. Ponovno tražimo dva simbola s najmanjim vjerojatnostima i kombiniramo ih u nad-simbol. Postupak se ponavlja sve dok binarno stablo nije do kraja konstruirano, odnosno dok nam ne ostanu samo vjerojatnosti za dva simbola, odnosno dok nam ne ostane samo jedan nad-simbol. Time kodiranje završava, a kod se

[16] Pandžić, I.S., Bažant, A., Ilić, Ž., Vrdoljak, Z., Kos, M., Sinković, V.: Uvod u teoriju informacije i kodiranje. Zagreb: Element d.o.o., 2007.., str. 255.

očitava prolazeći unatrag po binarnom stablu. Na kraju je svaki simbol kodiran zasebno, a kodne riječi nikako ne mogu imati manje od jednog bita po simbolu. Promotrimo stablo prikazano sljedećom slikom.



Slika 2. Postupak Huffmanovog kodiranja

Navedeno sada možemo i dodatno pojasniti pomoću stabla na slici. Prije početka konstruiranja stabla poznate su nam vjerojatnosti pojavljivanja simbola *a*, *b*, *c*, *d*, *e*, te su te vjerojatnosti naznačene u čvorovima koji ih predstavljaju. Vidimo da simbol *d* i simbol *e* imaju najmanje vjerojatnosti pa oni čine prva dva čvora koja konstruiramo te nakon toga stvaramo nad-simbol (u ovom slučaju to je čvor s vjerojatnosti 0.2 iz kojih se račvaju čvorovi simbola *d* i *e*). Unutar ovoga čvora upisan je zbroj vjerojatnosti simbola *d* i *e*, dok se na grane stabla upisuju brojevi 0 i 1. Zatim, gledajući i vjerojatnost tog nad-simbola i vjerojatnosti ostalih simbola, opet tražimo ona dva kod kojih su one najmanje te iz njih stvaramo novi nad-simbol, na isti način kao u koraku prije. Možemo uočiti na stablu da će ovaj nad-simbol imati vjerojatnost 0.4 što je veće od vjerojatnosti simbola *a* i *b*; zato ova dva simbola pišemo na istu razinu kao i simbol *c* i prvi nad-simbol, pošto je vjerojatnost sljedeće razine stabla veća od njihovih vjerojatnosti. Ovaj postupak

nastavljamo dok nam ne preostane samo jedan nad-simbol. U ovom slučaju to je čvor s vjerojatnosti 1. Sada vraćanjem kroz stablo odnosno od vrha prema dolje, s desna na lijevo, očitavamo naše kodove, a oni su sljedeći:

Simbol	Kodna riječ
a	00
b	01
c	10
d	110
e	111

Postupak Huffmanovog kodiranja može se sažeti u sljedećim koracima:[17]

1. Sortiramo simbole po padajućim vjerojatnostima.
2. Pronađemo dva simbola s najmanjim vjerojatnostima.
3. Jednom od njih dodijelimo 0, a drugom 1.
4. Kombiniramo ta dva simbola u jedan nad-simbol (dobije se zbrajanjem vjerojatnosti simbola od kojih je nastao) i zapišemo ga u čvor koji se račva na dva simbola iz kojih je nastao.
5. Ponavljamo prva četiri koraka dok ne dobijemo samo jedan nad-simbol.
6. Povratkom kroz stablo očitamo kodove.

Ovo je teoretski oblik algoritma nastao na inicijalnoj ideji kodiranja koji definira osnovu za provođenje postupka tako da traži optimalan kod. U praksi ovaj algoritam je obično malo izmijenjen kako bi odgovarao onome što se kodira i platformi na kojoj se kodira. Jedan izmijenjeni oblik ovog algoritma glasio ovako:[18]

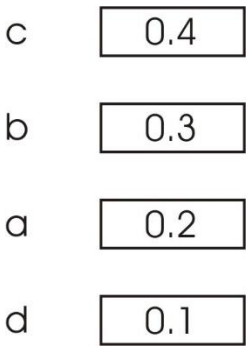
[17] Pandžić, I.S., Bažant, A., Ilić, Ž., Vrdoljak, Z., Kos, M., Sinković, V.: Uvod u teoriju informacije i kodiranje. Zagreb: Element d.o.o., 2007., str. 256.
[18] Muniswamy, V.V.: Design And Analysis Of Algorithms. New Delhi: I.K. International Publishing House Pvt. Ltd, 2009., str. 107.

1. Pronađi frekvenciju svakog simbola u datoteci koja se treba sažeti.
2. Za svaki simbol stvori binarno drvo s jednim čvorom koje sadrži taj simbol i njegovu frekvenciju kao prioritet.
3. Dodaj ovakva binarna drva u red prema prioritetima u redoslijedu sukladno rastućim frekvencijama.
4. *While (postoji više od jednog drveta u redu)*

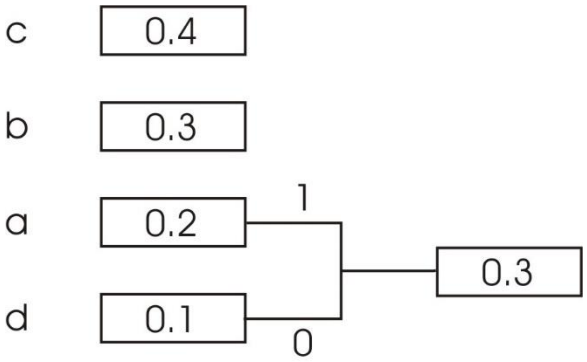
   //Makni dva drveta, t_1 i t_2 , iz reda

   //Stvori drvo t koje sadrži drvo t_1 kao lijevo pod-drvo i drvo t_2 kao desno pod-drvo
5. $PQ(t) = PQ(t_1) + PQ(t_2)$
6. *Umetni t na pripadajuću lokaciju u redu prema prioritetima*
7. *EndWhile*
8. Dodaj težine 0 i 1 na rubove drveta, tako da desni i lijevi rub drveta nemaju iste težine

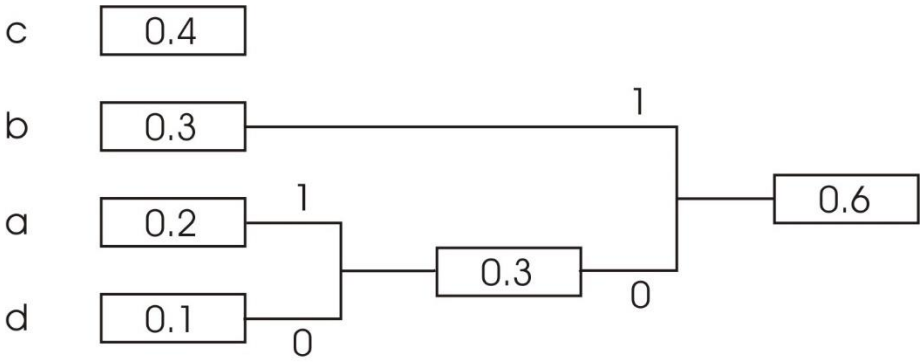
Postupak Huffmanovog kodiranja bit će objašnjen na jednostavnom primjeru u kojem je pretpostavljeno da su poznate vjerojatnosti pojavljivanja simbola *a*, *b*, *c* i *d*. Neka se pretpostavi da one iznose: *p*(a)=0.2, *p*(b)=0.3, *p*(c)=0.4 i *p*(d)=0.1. Prvo što moramo napraviti je poredati ih prema vjerojatnostima od najveće prema najmanjoj i to padajućim redoslijedom. Kada to napravimo dobivamo:



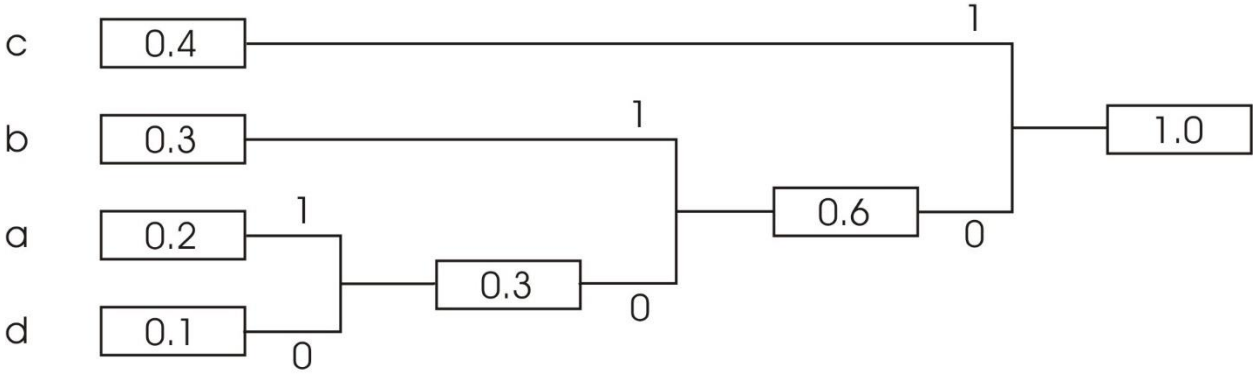
Sada dva simbola s najmanjim vjerojatnostima kombiniramo u jedan nad-simbol čija je vjerojatnost zbroj vjerojatnosti ta dva simbola. Na granama zapišemo brojeve 0 i 1.



Postupak ponavljamo dok nam ne ostane samo jedan nad-simbol. Dakle, u sljedećem koraku opet tražimo dva simbola s najmanjim vjerojatnostima, a to su simbol *b* i nad-simbol kreiran od simbola *a* i *d* u prvom koraku.



Na kraju je ostao samo jedan simbol i jedan nad-simbol. Za njih također ponovimo ranije korake.



Kada je postupak kodiranja završen, s desna na lijevo očitavamo kodne riječi pridružene simbolima:

Simbol	Kodna riječ
a	001
b	01
c	1
d	000

Simbolu s najvećom vjerojatnosti pojavljivanja (*c*) dodijeljena je najkraća kodna riječ, dok su simbolima s najmanjim vjerojatnostima pojavljivanja (*a* i *d*) dodijeljene kodne riječi najveće duljine. Time su ispunjene pretpostavke optimalnog koda. Ako bi sada pomoću ove tablice željeli kodirati poruku *aca*, pripadajući kod bi glasio 0011001. Da bi se poruka mogla kodirati, koderu signala mora biti poznata ova tablica, odnosno dodijeljene kodne riječi, a poruka će moći biti dekodirana samo u dekoderu signala koji poznaje istu tablicu.

4.2. Prednosti i nedostaci Huffmanovog kodiranja

Najveća prednost Huffmanovog kodiranja je njegova jednostavnost. Kreiranje binarnog stabla i očitavanje kodova je postupak koji se brzo uči, lako primjenjuje te ne zahtijeva previše razmišljanja. No, samo kodiranje uvelike ovisi o vjerojatnostima pojavljivanja simbola. Jedan je od nedostataka Huffmanovog kodiranja i taj što vjerojatnosti moraju biti unaprijed poznate kako bi se simboli mogli kodirati.

Svojstvo je Huffmanovog kodiranja da postoji „idealan raspored“ vjerojatnosti pojavljivanja simbola za koji je efikasnost koda jednaka 1. Naime, za niz simbola s vjerojatnostima pojavljivanja 1/2, 1/4, …, $1/2^n$, $1/2^n$ dobiva da je prosječna duljina kodne riječi jednaka entropiji. U stvarnosti to uglavnom nije slučaj. Kada vjerojatnosti nisu idealno raspoređene, a poglavito kada je jedna od vjerojatnosti izrazito velika (npr. 0.03, 0.07, 0.9), entropija je vrlo mala, jer nema puno neizvjesnosti. No, primjenom Huffmanovog kodiranja veliki broj ponavljanja simbola s najvećom vjerojatnosti dovodi do velike duljine kodirane poruke, budući da ne postoji način njenog sažimanja (Huffmanovo kodiranje ne dozvoljava manje od 1 bita po simbolu). Iako

je tako dobiven kod i dalje optimalan, njegova efikasnost nije velika. Ovaj problem može se riješiti primjenom proširenog Huffmanovog koda.

4.3. Prošireni Huffmanov kod

Da bi mogli kodirati i niz nepovoljnih vjerojatnosti nužno je uvesti neke preinake u osnovnom načinu kodiranja. Upravo na ovaj način se dobije prošireni Huffmanov kod. Osnovna preinaka koja se uvodi odnosi se na same vjerojatnosti jer se one kombiniraju međusobno i sve moguće kombinacije simbola predstavljaju osnovne simbole s kojima se dalje radi. Vjerojatnosti takvih kombiniranih simbola se dobiju međusobnim množenjem vjerojatnosti početnih simbola. Ova metoda otklanja problem lošeg rasporeda vjerojatnosti. Na taj se način dolazi do novog niza vjerojatnosti koji je primjereniji primjeni Huffmanovog kodiranja te ne daje tolika odstupanja prosječne duljine koda od entropije. Primjenom proširenog koda od n osnovnih vjerojatnosti dobivamo n^m kombiniranih vjerojatnosti i pripadajućih kodnih riječi.

Uzmimo sada opet vjerojatnosti 0.03, 0.07 i 0.9 i dodijelimo ih simbolima *A*, *B* i *C*, te primijenimo na njih metodu proširenog koda. Prvo tražimo sve moguće kombinacije od početne 3 vjerojatnosti za tri simbola u grupama po 2 vjerojatnosti što nam daje 3^2 mogućnosti, što je ukupno 9 novih vjerojatnosti koje postaju naš osnovni skup s kojim dalje radimo. One su redom:

Simbol	Vjerojatnost	Kodna riječ
AA	0.0009	00000000
AB	0.0021	0000001
AC	0.027	00001
BB	0.0049	000001
BC	0.063	001
CC	0.81	1
BA	0.0021	00000001
CA	0.027	0001
CB	0.063	01

Sada smo dobili novi niz vjerojatnosti pomoću kojih u sljedećem koraku možemo graditi binarno stablo. Primjećujemo da najveća vjerojatnost sada iznosi 0.81 za kombinaciju simbola *CC* što je ipak idealnije od vjerojatnosti 0.9 koju je imao početni simbol *C* pošto će rezultirati kraćom kodnom riječi i samim time manjom razlikom u odnosu na entropiju. Ovo nas dovodi do zaključka da je kodiranja kombiniranjem simbola, odnosno kodiranje blokova simbola, učinkovitije od kodiranja zasebnih simbola. Učinkovitost je još veća ako uzmemo veće blokove, odnosno kombiniramo simbole u grupe po tri ili četiri. No, što je više simbola u grupi broj rezultirajućih blokova raste eksponencijalno pa ova metoda ima granicu smislene upotrebljivosti.

4.4. Primjer Huffmanovog kodiranja pomoću PHP programskog jezika

Veći broj simbola značajno komplicira determiniranje optimalnog koda pomoću Huffmanove metode kodiranja. Upotrebom računala taj se postupak značajno pojednostavljuje. U nastavku je prikazan primjer Huffmanovog kodiranja pomoću PHP programskog jezika:[19]

```php
<?php
function encode($symb2freq) {
    $heap = new SplPriorityQueue;
    $heap->setExtractFlags(SplPriorityQueue::EXTR_BOTH);
    foreach ($symb2freq as $sym => $wt)
        $heap->insert(array($sym => ''), -$wt);

    while ($heap->count() > 1) {
        $lo = $heap->extract();
        $hi = $heap->extract();
        foreach ($lo['data'] as &$x)
            $x = '0'.$x;
        foreach ($hi['data'] as &$x)
            $x = '1'.$x;
        $heap->insert($lo['data'] + $hi['data'],
                      $lo['priority'] + $hi['priority']);
    }
    $result = $heap->extract();
```

[19] RosettaCode.org: Huffman Coding. URL: http://rosettacode.org/wiki/Huffman_coding (07.09.2016.)

```
    return $result['data'];
}

$txt = 'this is an example for huffman encoding';
$symb2freq = array_count_values(str_split($txt));
$huff = encode($symb2freq);
echo "Symbol\tWeight\tHuffman Code\n";
foreach ($huff as $sym => $code)
    echo "$sym\t$symb2freq[$sym]\t$code\n";
?>
```

Output koji se dobije pomoću ovog algoritma glasi:

Symbol Weight Huffman Code n 4 000 m 2 0010 o 2 0011 t 1 01000 g 1 01001 x 1 01010 u 1 01011 s 2 0110 c 1 01110 d 1 01111 p 1 10000 l 1 10001 a 3 1001 6 101 f 3 1100 i 3 1101 r 1 11100 h 2 11101 e 3 1111

Prvi dio outputa prikazuje simbole, drugi njihove težine (učestalosti pojavljivanja), a treći predstavlja sam Huffmanov kod. Kako bi se dobio jasniji prikaz kodnih riječi dodijeljenih simbolima, potrebno je posljednju liniju, koja u ovom slučaju ispisuje rezultate u nizu, prilagoditi tako da daje tablični ispis.

Samo kodiranje provodi funkcija *encode* koja se deklarira unutar koda. Funkcija koristi *build in* funkciju *SplPriorityQueue* kako bi odredila prioritet u kodiranju ovisno o frekvenciji pojavljivanja simbola. Drugi dio funkcije provodi samo kodiranje. Poruka koja se kodira spremljena je u varijablu *$txt*. Sama poruka se razlaže na simbole funkcijom *str_split* i određuje se njihov broj pojavljivanja, odnosno težina funkcijom *array_count_values*, pošto funkcija *str_split* vraća niz (*array*) vrijednosti. I sam rezultat funkcije *array_count_values* je niz oblika *$sym => $weight*, odnosno simbol => težina. Ova mreža je onda varijabla funkcije *encode* koja kodira simbole. Rezultat provođenja ove funkcije je također niz koji se sprema u varijablu *$huff*, a oblika je symbol => code. Na kraju se sve varijable šalju na ispis i dobije se gore navedeni output.

5. ARITMETIČKO KODIRANJE

Osim Huffmanovog kodiranja, aritmetičko kodiranje je jedan od najpoznatijih algoritama temeljenih na statističkim svojstvima izvora. Ova metoda predstavlja ulazni tekst kao broj između 0 i 1. Naknadni simboli dijele prije definirani interval ovisno o njihovim vjerojatnostima. Simboli s manjom vjerojatnosti značajno smanjuju duljinu intervala i na taj način daju više bitova prikazu, dok simboli s većom vjerojatnosti manje smanjuju duljinu intervala.[20]

Aritmetičko kodiranje, koje se temelji na kodiranju intervala, bitno je drugačije od Huffmanovog kodiranja, koje se temelji na kodiranju pojedinačnih simbola. Tako ono prevladava većinu nedostataka Huffmanovog kodiranja, poput kodiranja s najmanje jednim bitom po simbolu. Niz simbola izvorne abecede kodira se nizom simbola iz kodne abecede i tako se postiže kodiranje cijelog niza simbola, a ne samo pojedinačnih simbola.[21] Aritmetičko kodiranje često se primjenjuje pri kompresiji teksta, a model se sastoji od vjerojatnosti pojedinih simbola u nekom kontekstu. Najjednostavniji model koristi sveukupne frekvencije simbola u nekoj datoteci kao vjerojatnosti. Vjerojatnosti mogu biti procijenjene prilagodljivo tako da se počne od 1 za svaki simbol i onda se povećava nakon što je simbol kodiran ili na način da se broj simbola kodira prije kodiranja same datoteke te se modificira tijekom kodiranja ili se ostavi nepromijenjen. U svakom slučaju duljina kodne riječi je neovisna o redoslijedu simbola.[22]

5.1. Postupak kodiranja

Aritmetičko kodiranje generira kodnu riječ za određeni podinterval u ovisnosti o vjerojatnosti pojavljivanja pojedinačnih simbola od kojih je ta poruka sastavljena. Na taj se način direktno oslanja na entropiju samoga izvora na kojem simboli nastaju. To npr. znači da će neki simbol s većom vjerojatnosti pojavljivanja unutar aritmetičkog kodiranja predstavljati veći podinterval unutar početnog intervala [0, 1) dok će oni s manjom frekvencijom biti prikazani manjim

[20] Leondes, C.T. (ur.): Database and Data Communication Network Systems:Techniques and Applications, Volume 1. Amsterdam: Academic Press., 2002., str. 250-251.
[21] Shi, Q.J., Sun, H.: Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms and Standards. Boca Raton: CRC Press, 2008., str. 129.
[22] Kao, M.-Y. (ur): Encyclopedia of Algorithms. New York: Springer, 2008., str. 67.

podintervalom. Dakle, aritmetičko kodiranje dijeli početni interval [0, 1) na *n* podintervala, ovisno o broju simbola koji se pojavljuju na izvoru te o frekvenciji pojedinog simbola. Nadalje, same frekvencije predstavljaju podintervale koji će nastati podjelom. Ako npr. imamo izvor koji emitira 4 simbola A, B, C i D, s vjerojatnostima $p(A)=0.3$, $p(B)=0.2$ $p(C)=0.1$ i $p(D)=0.4$, dobivamo sljedeće podintervale: [0, 0.3), [0.3, 0.5), [0.5, 0.6) i [0.6, 1), odnosno širina svakog podintervala odgovora frekvenciji pojedinog simbola.

Jorma J. Rissanen

Jorma J. Rissanen rođen je 20. listopada 1932. godine u Lieksai, Finska. Jedan je od autora duljine minimalnog opisa kao novog principa u strojnom učenju. Provodio je značajna istraživanja, između ostalog, i na područjima predviđanja i teorije sustava, numeričke matematike, teorije vjerojatnosti i statistike. Radio je u IBM centru za istraživanje i nakon 30 godina rada trenutno je u mirovini. Također je i profesor emeritus na Sveučilištu za tehnologiju u Temperi. Tijekom svoje karijere objavio je veliki broj radova.[23]

Ako npr. želimo kodirati poruku AD dalje ćemo dijeliti onaj interval koji predstavlja prvi simbol naše poruke, a to je interval [0, 0.3). Ovaj interval ćemo opet podijeliti na isti broj podintervala na koji smo dijelili početni interval [0, 1), odnosno na 4 podintervala. Razlog je tome što to odgovara broju simbola iz početne poruke. Svaki novi interval dalje dijelimo na isti broj podintervala na koji smo dijelili početni interval [0, 1). Vratimo se na našu poruku koja glasi AD i koju želimo kodirati uporabom aritmetičkog kodiranja. Sada interval [0, 0.3) odnosno interval A dijelimo na 4 podintervala. Oni će onda izgledati ovako: [0, 0.09), [0.09, 0,15), [0.15, 0.18), [0.18, 0.3). Ovim podintervalima također pridružujemo slova iz abecede sukladno prvoj podjeli koju smo napravili na početku, odnosno simbole A, B, C i D. Tako smo zapravo dobili četiri nove poruke AA, AB, AC i AD iz prvog podintervala, odnosno podintervala A. Vidimo dakle da poruci AD odgovara podinterval [0.18, 0.3). Kao kodnu riječ za npr. poruku AB možemo uzeti bilo koju vrijednost iz intervala [0.09, 0.15) prikazanu u binarnom obliku.

[23] Wikipedia: Jorma Rissanen URL: https://en.wikipedia.org/wiki/Jorma_Rissanen (07.09.2016.)

Pretvaranje cijelog dekadskog broja u binarni oblik

Dekadski broj pretvaramo u binarni metodom količnika i ostatka. To radimo tako da broj podijelimo s 2 te količnike koje dobijemo nastavljamo dijeliti s 2 sve dok kao rezultat dijeljenja ne dobijemo 0. Ostatak dijeljenja može biti ili 0 ili 1, te taj ostatak zapišemo. Kada na posljetku, kao rezultat dijeljenja, dobijemo 0, zapisane ostatke čitamo od dolje prema gore, te tako dolazimo do vrijednosti dekadskog broja u binarnom obliku.

Kao primjer zapišimo broj 37 u binarnom obliku:

37 : 2 = 18	*uz ostatak*	*1*
18 : 2 = 9	*uz ostatak*	*0*
9 : 2 = 4	*uz ostatak*	*1*
4 : 2 = 2	*uz ostatak*	*0*
2 : 2 = 1	*uz ostatak*	*0*
1: 2 = 0	*uz ostatak*	*1*

Dekadski broj 37 zapisan u binarnom sustavu glasi 100101.

Pretvaranje dekadskog broja manjeg od 1 u binarni oblik

Brojevi manji od 1 pretvaraju se u binarni oblik tako da se množe s brojem 2 te se tako formira niz cjelobrojnog viška. Kao primjer zapišimo broj 0.25 u binarnom obliku:

0.25 x 2	*=*	***0.5***	*=*	***0.5***	*+*	***0***
0.5 x 2	*=*	*1*	*=*	*0*	*+*	*1*

Čitajući od gore prema dolje zadnje znamenke u svakom redu dobijemo da binarni zapis dekadskog broja 0.25 koji glasi 0.01. Ako kao rezultat množenja dobijemo broj veći od 1 uzimamo ostatak do 1 (npr. ako je rezultat množenja s 2 broj 1.65, pišemo 0.65 + 1, gdje je 0.65 ostatak, a 1 znamenka binarnog zapisa. Dalje bi onda ostatak množili s 2 (0.65 x 2), itd.

Poruci AD možemo pridružiti dekadski broj 0.25, budući da se nalazi u intervalu [0.18, 0.3). Binarni zapis tog broj glasi 0.01 pa poruku AD kodiramo kao 01. Dakle, kodnu riječ biramo iz podintervala kojem pripada poruka koju želimo kodirati. Tu poruku možemo kodirati s binarnim

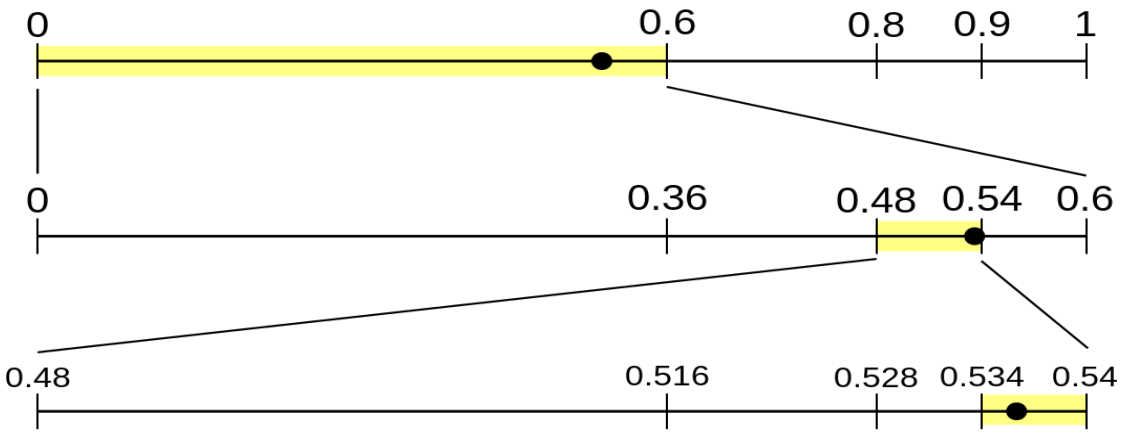
zapisom bilo koje vrijednosti iz pripadajućeg podintervala. Ako želimo kodirati neku drugu poruku, npr. ABC, moramo ponoviti postupak tako da interval AB podijelimo na 4 nova podintervala kako bi dobili odgovarajući podinterval za poruku ABC. Želimo li kodirati poruku ABCD moramo dalje dijeliti interval ABC na 4 nova podintervala te u podintervalu ABCD tražili kodnu riječ.

U skladu s navedenim, možemo definirati 5 glavnih koraka aritmetičkog kodiranja[24]:

1. Interval [0, 1) podijelimo na *n* podintervala koji odgovaraju slovima abecede (*n* ovisi o broju simbola na izvoru poruke).
2. Iz promatranog skupa podintervala odaberemo onaj koji odgovara sljedećem simbolu u poruci.
3. Taj podinterval opet podijelimo na *n* novih podintervala proporcionalno vjerojatnostima pojavljivanja simbola iza abecede. Tako nastaje novi skup podintervala.
4. Ponavljamo korake 2 i 3 dok cijela poruka nije kodirana.
5. Konačni kod za poruku je jedan broj iz odgovarajućeg intervala zapisan u binarnom obliku.

Svaki sljedeći interval u poruci uvijek je jednoznačno određen prethodnim intervalima, odnosno prethodnim simbolima u nizu. Postupak aritmetičkog kodiranja simbolički se može prikazati sljedećom slikom. Na slici se može uočiti podjela intervala na odgovarajuće podintervale.

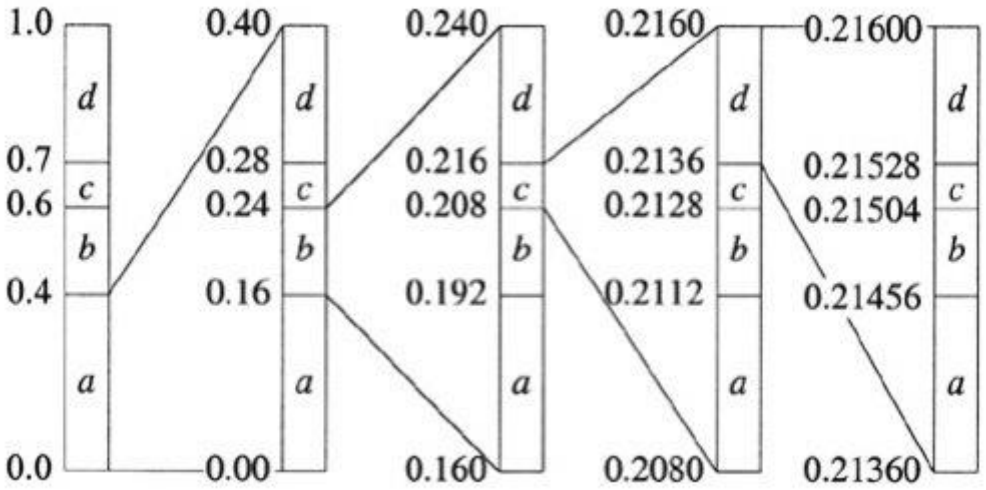
[24] Pandžić, I.S., Bažant, A., Ilić, Ž., Vrdoljak, Z., Kos, M., Sinković, V.: Uvod u teoriju informacije i kodiranje. Zagreb: Element d.o.o., 2007., str. 261.



Slika 3. Shema aritmetičkog kodiranja

Izvor: Wikipedia: Arithmetic coding. URL: http://en.wikipedia.org/wiki/Arithmetic_coding (07.09.2016.)

Objasnimo postupak aritmetičkog kodiranja na još jednom primjeru. Pretpostavimo da neki izvor emitira 4 simbola *a*, *b*, *c* i *d*, s vjerojatnostima *p*(a)=0.4, *p*(b)=0.2, *p*(c)=0.1 i *p*(d)=0.3. Neka se nadalje pretpostavi da je zadatak odrediti kodnu riječ za poruku *abcd*. Sljedeća slika prikazuje način na koji se dolazi do tražene kodne riječi.



Slika 4. Postupak aritmetičkog kodiranja

Izvor**:** Müller, P.: Entropy Coding. URL: http://www.icsy.de/studium/vorlesung/ws0910_MMS/pdf/05-compression-02.pdf (07.09.2016.)

Podijelivši početni interval na odgovarajući broj podintervala dobili smo interval koji se odnosi na poruku *abcd*. To je interval [0.2136, 0.2160). Sada iz ovog intervala odabiremo bilo koju

vrijednost te je pretvaramo u binarni oblik kako bi dobili kodnu riječ. Pretpostavimo da smo odabrali broj 0.21484375. Postupak pretvaranja je sljedeći:

0.21484375 x 2	=	0.4296875	=	0.4296875	+	0
0.4296875 x 2	=	0.859375	=	0.859375	+	0
0.859375 x 2	=	1.71875	=	0.71875	+	1
0.71875 x 2	=	1.4375	=	0.4375	+	1
0.4375 x 2	=	0.875	=	0.875	+	0
0.875 x 2	=	1.75	=	0.75	+	1
0.75 x 2	=	1.5	=	0.5	+	1
0.5 x 2	=	1	=	0	+	1

Na kraju zapišemo niz znamenki s krajnje desne strane od gore prema dolje. Tako dobivamo 0.00110111, što predstavlja binarni zapis broja 0.21484375. Prema tome, tražena kodna riječ poruke *abcd* glasi 00110111 (zanemaruje se 0 ispred decimalne točke).

Dobro je promisliti pri odabiru vrijednosti iz koje ćemo izvesti kodnu riječ jer se tako može značajno skratiti kodna riječ. Poželjno je birati onu vrijednost koja se može najjednostavnije pretvoriti u binarni broj. Takva je npr. vrijednost čiji se brojnik može zapisati kao potencija s bazom.

5.2. Postupak dekodiranja

Za dekodiranje kodne riječi nastale aritmetičkim kodiranjem potrebne su nam vjerojatnosti pojavljivanja simbola na izvoru te kodna riječ koju smo dobili kodiranjem željene poruke. Postupak dekodiranja je sličan postupku kodiranja, jedino što ovaj put dobivenu kodnu riječ u obliku binarnog broja pretvaramo u dekadski te tražimo interval u kojemu se taj broj nalazi. Kada nađemo odgovarajući interval samo očitamo simbole poruke koji korespondiraju s intervalom. Postupak dekodiranje odvija se na sljedeći način:[25]

[25] Pandžić, I.S., Bažant, A., Ilić, Ž., Vrdoljak, Z., Kos, M., Sinković, V.: Uvod u teoriju informacije i kodiranje. Zagreb: Element d.o.o., 2007., str. 263.

1. Podijelimo početni interval [0, 1) u podintervale na temelju vjerojatnosti pojavljivanja simbola.
2. Dobivenu kodnu riječ zapisanu u obliku binarnog broja pretvorimo u dekadski broj.
3. Determiniramo podinterval u kojemu se nalazi dobiveni kod.
4. Zapišemo simbol koji odgovara tom podintervalu.
5. Taj podinterval dijelimo na novih *n* podintervala (*n* odgovara broju simbola na izvoru signala) proporcionalno vjerojatnostima pojavljivanja simbola iz abecede.
6. Ponavljamo korake 3 do 5 dok ne dođemo do kraja poruke.

Pri dekodiranju poruke javlja se jedan problem. Naime, pitanje je kako znamo kada je kraj poruke. Bez dodatnog obilježavanje to nije moguće utvrditi pa se ovaj problem rješava dodavanjem posebnog simbola za kraj poruke. Tako primatelj poruke može znati kada treba stati s dekodiranjem te se time otklanja potencijalna beskonačna petlja.

Pretpostavimo da treba dekodirati poruku iz prethodnog primjera čija je kodna riječ 00110111, odnosno zapisano s decimalnom točkom 0. 00110111.

Pretvaranje binarnog broja u dekadski oblik

Binarni broj pretvaramo u dekadski oblik prema sljedećoj formuli:

$$N_{10} = a \times 2^n + \cdots + a \times 2^2 + a \times 2^1 + a \times 2^0$$

gdje je a znamenka sustava (0 ili 1), a n-1 broj binarnih znamenki.

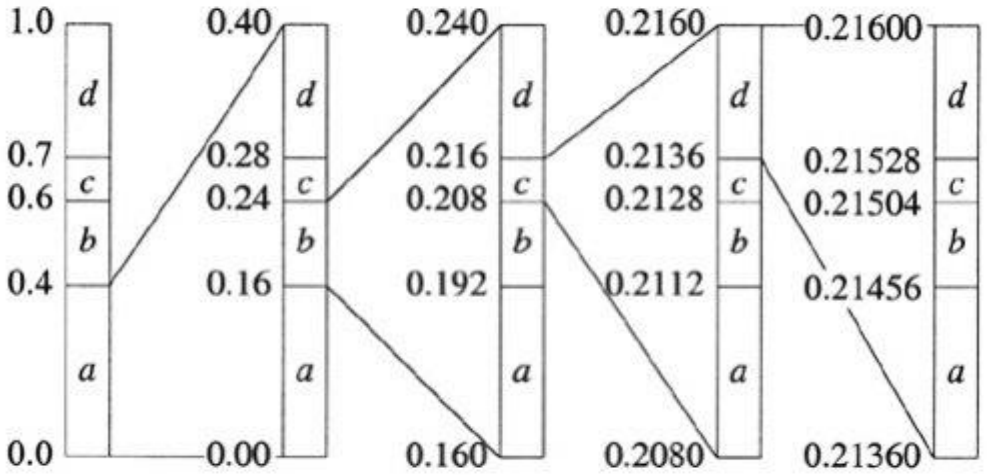
Pretvaranje binarnog broja manjeg od 1 u dekadski oblik

Binarni broj manji od 1 pretvaramo u dekadski oblik raspisivanjem po negativnim potencijama baze 2:

$$N_{10} = a \times 2^{-1} + a \times 2^{-2} + a \times 2^{-3} + \cdots + a \times 2^{-n}$$

gdje je a znamenka sustava (0 ili 1), a n broj binarnih znamenki iza decimalne točke.

Pretvoreno u dekadski oblik to je broj 0.21484375. Prvo moramo podijeliti interval [0, 1) na četiri podintervala te tim podintervalima pridružiti slova iz abecede. Zatim odredimo u kojem se podintervalu nalazi dobiveni kodni broj. U ovom slučaju to je prvi podinterval pa zato zapisujemo slovo *a*. Zatim taj podinterval dijelimo na 4 nova podintervala te tražimo u kojem se sada podintervalu nalazi naš kodni broj te zapišemo odgovarajući simbol i taj podinterval opet dijelimo u 4 nova podintervala. Ovaj postupak nastavljamo sve dok ne dođe kraj poruke.



Slika 5. Postupak aritmetičkog dekodiranja

Izvor**:** Müller, P.: Entropy Coding. URL: http://www.icsy.de/studium/vorlesung/ws0910_MMS/pdf/05-compression-02.pdf (07.09.2016.)

Iz ove slike može se jasno zaključiti da poruka glasi *abcd*, čime je postupak dekodiranja završen.

5.3. Problemi aritmetičkog kodiranja

Nekoliko je osnovnih problema koji se javljaju kod primjene aritmetičkog kodiranja. Zato je opisani algoritam prvenstveno upotrebljiv samo u teoretskom smislu, dok se u praksi ne može uspješno primijeniti. Razlog za ovo leži u nizu realnih pretpostavki koje izvorni algoritam ne može zadovoljiti. Tako se kao prvi problem javlja potreba da bude poznata cijela poruka kako bi započeo postupak kodiranja, što onemogućava komunikaciju u realnom vremenu. Drugi problem leži u potrebi za realnim brojevima s većim brojem decimalnih mjesta do koje dolazi uslijed sve veće podjele podintervala. Zbog toga na posljetku dolazi do premašivanja preciznosti koju podržava zapis na računalu. Treći problem je u procesorskoj snazi koja je potreban za rad s

decimalnim brojevima koja postaje veća s povećanjem decimalnih mjesta, što se onda odražava na vrijeme potrebno za procesiranje. Kako bi se riješili ovi problemi potrebno je kreirati prilagođene algoritme koji bi zadovoljavali sljedeća tri uvjeta:[26]

1. Korištenje operacija s cijelim brojevima.
2. Korištenje prikaza s fiksnim brojem bitova.
3. Proizvodnja simbola koda tijekom postupka kodiranja, a ne tek na kraju poruke.

Jedan od načina koji zadovoljava ove uvijete je metoda renormalizacije intervala. Ona slijedi opisani algoritam za kodiranje, ali uvodi i neke promjene. Prva od njih je da se za prikaz gornje i donje granice intervala koristi fiksni broj znamenki. Druga promjena odnosi se na simultano kodiranje poruke, odnosno ako se desi da su prvih *n* znamenki gornje i donje granice jednake, one se odmah šalju na izlaz kodera te postaju dio kodne riječi, a sve ostale znamenke u granici oba intervala se pomiču za jedno mjestu u lijevo, dok se na desnoj strani dodaju znamenke. Tako se poruka istovremeno kodira kako se izvodi, što eliminira potrebu za čekanjem kraja poruke kako bi se kodirala. Ova prilagodba omogućuje praktičnu upotrebu algoritamskog koda.

5.4. Prednosti i nedostaci aritmetičkog kodiranja

Osnovna prednost aritmetičkog kodiranja je njegova mogućnost da kodira cijelu poruku jednim kodom koji se onda uzima iz odgovarajućeg intervala. Druga prednost je u količini podataka potrebnih za kodiranje, odnosno ekvivalentno tome o kompresiji koja je moguća. Aritmetičko kodiranje omogućuje da se kodira s manje od 1 bita po simbolu, čime se postiže bolja kompresija te se zauzima manje memorije, nego u slučaju Huffmanovog kodiranja. Samim time se dobivaju bolji rezultat za dulje poruke. Dva glavna nedostatka, uz potrebu za modifikacijom pri implementaciji, su zahtjevnost samog koda koja je u usporedbi s drugim metodama entropijskog kodiranja ipak nešto veća te patent koji postoji za ovaj algoritam, zbog čega je za korištenje potrebno otkupljivanje licence.

[26] Pandžić, I.S., Bažant, A., Ilić, Ž., Vrdoljak, Z., Kos, M., Sinković, V.: Uvod u teoriju informacije i kodiranje. Zagreb: Element d.o.o., 2007., str. 265.

5.5. Primjer upotrebe aritmetičkog kodiranja pri kompresiji slike

Jedna o praktičnih uporaba aritmetičkog kodiranja je pri kompresiji slike. Metoda koja se tu primjenjuje slijedi nakon statističke analize slike pri kojoj se slika razlaže na elemente koji je čine i koji se pojavljuju s različitim učestalostima. Samim time te diskretne vrijednosti elemenata imaju određenu vjerojatnost pojavljivanja što zapravo definira temeljne vjerojatnosti s kojima se provodi kodiranje. Pretpostavimo da neka slika ima elemente koji mogu poprimiti vrijednosti iz skupa S = {A, B, C, D, E} te da se elementi na slici pojavljuju sa sljedećim frekvencijama A = 100, B = 200, C = 400 D = 100 i E = 600, što onda daje vjerojatnosti pojavljivanja simbola *p*(A)=0.07, *p*(B)=0.14, *p*(C)=0.29, *p*(D)=0.07 i *p*(E)=0.43. Dalje se kodiranje povodi istim algoritmom koji smo pokazali ranije. Razlog zbog kojega je algoritamsko kodiranje ujedno i vrlo efikasan način kompresije je što ono raspon vjerojatnosti od 0 do 1 smanjuje na puno manji raspon i time reducira broj bitova potrebnih za zapis i kompresiju. Upravo takav način kodiranja omogućava bolju kompresiju metodom aritmetičkog kodiranja, nego npr. metodom Huffmanovog kodiranja.

Entropijsko kodiranje u JPEG standardu provodi se u dva koraka. Prvo se koeficijenti pretvaraju u intermedijalnu sekvencu simbola koji se onda kodiraju primjenom Huffmanovog ili aritmetičkog kodiranja.[27] Drugi korak se odnosi na različito kodiranje različitih koeficijenata.

Osnovni ciljevi JPEG standarda su sljedeći:[28]

1. Postići stupanj rekonstrukcije kvalitete slike koja je točna ili približna stanju umjetničkog djela, pri čemu se vjernost slike može okarakterizirati kao vrlo dobra ili odlična.
2. Biti koristan za kompresiju gotovo bilo kojeg oblika slike, uključujući sivu paletu (crno – bijelo) ili bilo koju paletu boja i većinu veličina slika.
3. Imati kompleksnost koja dopušta implementaciju na mnogo računalnih platformi s dostupnom hardverskom implementacijom.

[27] Gibson, J.D., Berger, T., Lookabaugh, T., Lindbergh, D., Baker, R.L.: Digital Compression for Multimedia: Principles and Standards. Amsterdam: Morgan Kauffman Publishers, 1998., str. 291.
[28] Gibson, J.D., Berger, T., Lookabaugh, T., Lindbergh, D., Baker, R.L.: Digital Compression for Multimedia: Principles and Standards. Amsterdam: Morgan Kauffman Publishers, 1998., str. 301.

Većina ovih ciljeva postiže se upravo korištenjem metoda kompresije bez gubitaka te upravo primjenom Huffmanovog i/ili aritmetičkog kodiranja.

6. METODE RJEČNIKA

Metode rječnika, kako im ime i govori, održavaju i koriste rječnik ili knjigu kodova za riječi ili nizove simbola koji su se prije pojavljivali u korištenom tekstu, odnosno u prijašnjim unosima, pri čemu se kompresija postiže mijenjanjem nizova u tekstu s referencama iz rječnika. Sam rječnik je prilagodljiv i dinamičan u smislu da se stvara dodavanjem novih nizova simbola te dopušta brisanje nizova simbola koji se rjeđe koriste ili se više uopće ne koriste ako veličina rječnika prijeđe maksimalni limit. Također je moguće koristi statični rječnik za kompresiju teksta pri čemu su nizovi simbola unaprijed upisani.[29]

Izvori s memorijom označavaju da vjerojatnosti pojavljivanja određenih simbola ovise o jednom ili više prethodnih simbola poruke te se mijenjaju ovisno o njima. Izvor s memorijom „pamti“ simbole koji su prethodili. Tako npr. ako su dva simbola, koja su prethodila, bila *i* i *l* onda veću vjerojatnost za pojavljivanje ima ponovno simbol *i* pošto se tako formira riječ *ili*, koju često susrećemo u tekstu. Isto tako, ako su se npr. redom pojavili simboli *a*, *u*, *t*, *o*, *m*, *o*, *b*, povećava se vjerojatnost da će dva sljedeća simbola biti *i* i *l*, što daje riječ automobil. Logično je primijetiti da u svakodnevnom govoru postoje riječi koje se češće pojavljuju od drugih što će utjecati na vjerojatnost pojavljivanja određenih simbola na izvorima s memorijom. Izvor će pamtiti simbole kako oni nastaju te će s vremenom ponavljati određene dijelove teksta, fraze ili čak i cijele riječi. Metode rječnika se upravo temelje na kodiranju ovih dijelova teksta tako da se pojedinačni skupovi simbola, koji se često pojavljuju, kodiraju kao jedan simbol, što povećava mogućnosti kompresije. Velika prednost ovakvog načina kodiranja u odnosu na Huffmanovo ili aritmetičko kodiranje je što vjerojatnosti pojavljivanja simbola na izvoru ne moraju biti poznate kako bi se simboli kodirali. Izvor jednostavno pamti simbole te se kodiraju određene skupine simbola koje se na njemu češće pojavljuju bez potrebe za znanjem točne vjerojatnosti tog pojavljivanja. Ovi novi skupovi simbola koji nastanu unose se u rječnik kodera signala. Logično, nužno je da se unesu i u rječnik dekodera kako bi se signal mogao dekodirati. Na ovaj način nastaju dinamički rječnici koji se automatski prilagođavaju novim unosima. Druga opcija je stvaranje statičkog rječnika u kojemu su grupe simbola unaprijed poznate i unesene te se ne mogu mijenjati. U praksi su efikasniji dinamički rječnici te se oni uglavnom i koriste. Zbog svoje mogućnosti

[29] Sayood, K. (ur): Lossless Compression Handbook. Amsterdam: Academic Press, 2003., str. 232.

prilagođavanja novim unosima, dinamički rječnici nazivaju se još i univerzalnim rječnicima. U nastavku su izložena tri algoritma za kodiranje metodom rječnika: LZ77, LZ78 i LZW.

6.1. LZ77

Ovaj algoritam kodira poruku koristeći sljedeća četiri alata: posmični prozor, prozor za kodiranje, pomak i duljinu riječi te sljedeći simbol, a koristi ih na sljedeći način. Posmični prozor se sastoji od *N* prethodno kodiranih simbola poruke, a prozor za kodiranje sadrži *N* simbola koji se trebaju kodirati. Samo kodiranje se provodi tako da se u prozoru za kodiranje traži najdulji mogući niz simbola koji se može pronaći u posmičnom prozoru te se taj niz sljedeći kodira. Kada je niz pronađen određuju se duljina riječi i pomak. Pomak je udaljenost od prvog simbola riječi u posmičnom prozoru, identične onoj koja se sljedeća kodira, i prvog simbola same riječi koja se kodira, uključujući i taj prvi simbol. Duljina je jednostavno duljina riječi koja se kodira. Sljedeći simbol je onaj koji slijedi nakon riječi koja je kodirana u *i*-tom koraku. U koraku $i+1$ posmični prozor pomiče se tako da zadnji simbol koji obuhvaća bude upravo posljednji simbol koraka *i*. Algoritam na koder signala šalje uređeni skup sljedeće tri vrijednosti (pomak, duljina, posljednji simbol). Kontinuiranim pomicanjem posmičnog prozora i prozora za kodiranje u $i+n$ koraka kodira se cijela poruka. Sama duljina posmičnog prozora i duljina prozora za kodiranje mogu se proizvoljno odrediti. U praksi je uobičajena duljina posmičnog prozora oko 2000 simbola. Upravo i postojanje prosječne duljine posmičnog prozora, odnosno kodiranje samo pomoću njega je ujedno i glavni nedostatak ove metode pošto algoritam ne pamti ništa što slijedi prije posmičnog prozora te može provoditi daljnje kodiranje samo temeljem simbola koji se nalaze u njemu. Ipak posmični prozor može biti veće duljine, čime se povećava broj simbola koje obuhvaća, odnosno povećava se memorija. Posljedično, to dovodi do usporavanja postupka jer iz perspektive računala zahtjeva više vremena za pretraživanje posmičnog prozora i zahtjeva više procesorske snage te više računalne memorije.

Abraham Lempel

Rođen je 10. veljače 1936. godine u današnjem Lvivu u Ukrajini. Studirao je na Izraelskom institutu za tehnologiju gdje je magistrirao 1963. godine, a doktorirao 1967. godine. Trenutno je profesor emeritus na navedenom sveučilištu. Njegov vjerojatno najznačajniji rad je nastao u suradnji s Jacobom Zivom, koja je rezultirala kreiranjem Lempel-Ziv algoritma za kompresiju bez gubitaka.[30]

Navedeno možemo ilustrirati na jednom primjeru. Pretpostavimo da želimo kodirati riječ **pakokpakoapakapo** te uzmimo da će duljina našeg posmičnog prozora biti 5 simbola, kao i duljina prozora za kodiranje. Sada još pretpostavimo da smo već kodirali jedan dio poruke sve do drugog simbola k tako da ovom *i*-tom koraku s kojim krećemo u daljnje kodiranje imamo sljedeće:

p a k o k p a k o a p a k a p o ; pomak: 5 duljina: 4

Vidimo da u ovom koraku možemo naći riječ **pako** u posmičnom okviru (označen pravokutnikom s punom crtom) i okviru za kodiranju (isprekidana crta) i to je sljedeća riječ koja se kodira. Pomak je 5 simbola, a duljina riječi je 4 simbola. Dakle, ovaj korak na izlaz kodera šalje triplet (5, 4, a) pošto je *a* sljedeći simbol. Sada se oba okvira pomiču i sljedeći korak izgleda ovako:

p a k o k p a k o a p a k a p o ; pomak: 5 duljina: 3

Posmični okvir se pomaknuo pa ponavljamo postupak iz prethodnog koraka. Ovaj put najdulja riječ koja se pojavljuje u oba okvira je **pak** pa je triplet koji se šalje na izlaz kodera (5, 3, a), pošto je sljedeći simbol a.

p a k o k p a k o a p a k a p o ; pomak: 4 duljina: 1

[30] Wikipedia: Abraham Lempel. URL: https://en.wikipedia.org/wiki/Abraham_Lempel (07.09.2016.)

Dakle, u ovom koraku se kodira samo **p** pošto ne postoji šire podudaranje u posmičnom okviru i okviru za kodiranje. Triplet koji se šalje na izlaz kodera je (4, 1, o). Preostaje nam još jedino taj zadnji simbol koji je poslan koderu i time je postupak kodiranja završen. Rezultat kodiranja (od koraka kojim smo krenuli) je sljedeći :

Korak	Kod
1	(5, 4, a)
2	(5, 3, a)
3	(4, 1, o)

Sada kada samo naučili kako kodirati neku poruku metodom rječnika možemo pogledati što se dešava u dekoderu signala, odnosno kako se poruka dekodira. Pretpostavimo da je dio poruke **pakok** već dekodiran te na dekoder sljedeći stiže triplet (5, 4, a) što znači da s pozicije nakon zadnjeg simbola prijašnjeg koraka (koji pretpostavljamo) idemo unatrag 5 simbola i uzimamo 4 simbola, dakle **pako**, što je sljedeći dio naše poruke koja sada glasi **pakokpako**. Dodajemo još zadnji simbol i dobivamo **pakokpakoa.** Zatim na dekoder stiže sljedeći triplet (5, 3, a). Sada krećemo od zadnjeg simbola prošlog koraka, simbola *a*, i idemo unatrag 5 simbola i uzimamo 3 prva simbola čime dobijemo **pak.** Ovo opet dodajemo na našu poruku koja sada glasi **pakokpakoapaka** nakon što smo dodali i zadnji simbol *a*. Preostaje nam još triplet (4, 1, o) pomoću kojega dobijemo simbol **p** i dodajući još zadnji simbol *o* dobijemo cijelu poruku koja glasi **pakokpakoapakapo**, što je naša izvorna poruka.

Kao što je već napomenuto, glavni nedostatak ove metode je zadana duljina posmičnog okvira zbog čega dosta često, pogotovo ako je izvor periodičan s periodom malo duljim od posmičnog raspona, daje loše kodove.

6.2. LZ78

Ova metoda je zapravo unaprijeđena i poboljšana LZ77 metoda koja umjesto posmičnog okvira koristi rječnik. Sam rječnik je na početku prazan, a popunjava se tijekom kodiranja spremanjem novih nizova simbola koji se ponavljaju te se na taj način rječnik postupno nadopunjuje dok

kodiranje traje. Sama metoda kodiranja, osim korištenja rječnika, ne razlikuje se previše od LZ77 metode. Sam rječnik se koristi pomoću indeksiranja, koje se provodi tako da se u svakom koraku kodiranja na izlaz kodera šalju indeks u rječniku i kod sljedećeg simbola. Sam indeks je redni broj najdulje riječi u rječniku koja je jednak nadolazećoj poruci za kodiranje. Dakle, slično je metodi LZ77, ali ovaj put bez posmičnog okvira. Simbol koji slijedi se pojedinačno kodira i tako se dobije „sljedeći simbol" koji se koristi u metodi LZ77. Kako se kodiranje nastavlja rječnik se nadopunjuje novim kodovima. Sam rječnik naravno mora biti poznat i koderu i dekoderu signala kako bi se kodiranje i dekodiranje moglo uspješno provesti. Uvođenje rječnika nadoknađuje nedostatak koji postoji kod metode LZ77, pošto se eliminira posmični okvir, a sam rječnik je dinamičan te se stalno nadopunjuje.

Jacob Ziv

Rođen je u Tiberiasu u Palestini 27. studenog 1931. godine. Diplomirao je i magistrirao na Sveučilištu u Izraelu, a doktorirao na MIT-u. Radio je na Sveučilištu u Izraelu i u Bellovom istraživačkom laboratoriju. Područja su njegovog istraživanja teorija informacije, kompresija podataka i statistička teorija komunikacije. Njegov najpoznatiji rad je na Lempel–Ziv algoritmu, koji je razvio s Abrahamom Lempelom.[31]

6.3. LZW

LZW je unaprijeđena LZ78 metoda, a glavna razlika i prednost ove metode je postojanje unaprijed definiranog rječnika osnovnih simbola abecede što isključuje potrebu kodiranja posljednjeg simbola te se rječniku šalje samo indeks pozicije simbola. Sam rječnik se također nadopunjuje prilikom kodiranja, slično kao i kod LZ78 metode. Svaka nova riječ koja ne postoji u rječniku biti će u njega upisana te indeksirana i kao takva postati novi unos koji će rječnik kasnije moći koristiti. Što se tiče osnovnih unosa, izvorni rječnik LZW metode za prvih 256 riječi koristi standardne ASCII[32] kodove. Sam postupak kodiranja ove metode prati nekoliko jednostavnih koraka. Princip LZW kodiranja je u tome da koder unosi simbole jedan po jedan i akumulira ih u string I. Dokle god se string I može naći u rječniku proces se nastavlja. U nekom

[31] Wikipedia: Jacob Ziv. URL: https://en.wikipedia.org/wiki/Jacob_Ziv (07.09.2016.)
[32] Standardni američki znakovnik za razmjenu informacija

trenutku dodavanje novog simbola x prouzrokuje neuspješno pretraživanje rječnika pošto je niz simbola I u rječniku, ali simbol (ili niz simbola) x nije. U tom slučaju dolazi do sljedeće tri stvari:[33]

1. Koder radi output sa sadržajem niza simbola I.
2. Sprema niz simbola Ix u sljedeći dostupni unos u rječnik.
3. Dodaje niz I simbolu (ili nizu) x, odnosno x je novi I.

Oni se mogu zapisati u obliku sljedećeg pseudokoda:[34]

1. *RadnaRiječ = sljedeći simbol poruke na ulazu*
2. *WHILE (ima još simbola na ulazu) DO*
3. *NoviSimbol = sljedeći simbol s ulaza*
4. *IF RadnaRiječ + NoviSimbol postoji u rječniku THEN*
5. *RadnaRiječ = RadnaRiječ + NoviSmbol*
6. *ELSE*
7. *IZLAZ: kod za RadnaRiječ*
8. *dodaj (RadnaRiječ + NoviSimbol) u rječnik i dodijeli tom izrazu novi kod*
9. *RadnaRiječ = NoviSimbol*
10. *ENDIF*
11. *ENDWHILE*
12. *IZLAZ: kod za RadnaRiječ*

Algoritam na početku uzima prvi simbol u nizu, odnosno prvi simbol poruke i taj simbol postaje radna riječ. Zatim uzima sljedeći simbol poruke i s njime proširuje radnu riječ. Ovisno o tome postoji li ova nova riječ u rječniku, ona će, ako postoji, postati nova radna riječ, a ako ne postoji na izlaz kodera šalje se kod za radnu riječ dok se ova nova riječ zapisuje u rječnik i dodaje joj se novi kod. Konačno se za novu radnu riječ uzima sljedeći simbol poruke te se koraci od 2 do 11 ponavljaju.

[33] Salomon, D.: Data Compression: The Complete Reference, Fourth Edition. London: Springer, 2007., str. 199.
[34] Pandžić, I.S., Bažant, A., Ilić, Ž., Vrdoljak, Z., Kos, M., Sinković, V.: Uvod u teoriju informacije i kodiranje. Zagreb: Element d.o.o., 2007.., str. 272.

Terry Archer Welch

Terry Welch rođen je 20. siječnja 1939. godine, a umro je 22. studenog 1988. godine. Američki je znanstvenik koji je najpoznatiji po svom radu na kompresiji podataka. S Abrahamom Lempelom i Jacobom Zivom razvio je algoritam za kompresiju bez gubitaka koji je po njima dobio i ime Lempel-Ziv-Welch algoritam. Diplomirao je i doktorirao elektrotehniku na MIT-u, a radio je na Sveučilištu Texas.[35]

Promotrimo rečeno na primjeru i kodirajmo poruku *OPPOPOPOR* koristeći LZW metodu. Kako bi nam bilo lakše uočiti mjesta simbola napravit ćemo sljedeću tablicu.

MJESTO	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
SIMBOL	O	P	P	O	P	O	P	O	R

Neka rječnik na početku poznaje sljedeće simbole.

INDEKS	**SIMBOL**
(1)	O
(2)	P
(3)	R

Korake kodiranja možemo prikazati pomoću ove tablice.

KORAK	**MJESTO**	**SADRŽAJ RJEČNIKA**	**IZLAZ**
1	1	(4)OP	(1)
2	2	(5)PP	(2)
3	3	(6)PO	(2)
4	4	(7)OPO	(4)
5	6	(8)OPOR	(7)
6	9		(3)

[35] Wikipedia: Terry Welch. URL: https://en.wikipedia.org/wiki/Terry_Welch (07.09.2016.)

Na početku uzimamo prvi simbol poruke te nam on postaje radna riječ (u ovom slučaju to je simbol *O*). Simbol *O* postoji u rječniku te mu dodajemo sljedeći simbol iz poruke, a to je *P*, te tako nastaje *OP*. Ova riječ ne postoji u rječniku te je zapisujemo u njega i dodajemo joj sljedeći slobodan kod, a to je (4), te za radnu riječ uzimamo sljedeći simbol poruke (*P*). Sada ponavljamo korake algoritma. Dakle, ponovo gledamo postoji li *RadnaRiječ + Novi simbol* (ovdje je to *PP*) u rječniku. Budući da ne postoji, dodajemo je u rječnik i pridružujemo novi kod (5). Zatim uzimamo sljedeći simbol za novu radnu riječ, a to je *P*. Postupak za treći korak je isti kao i za drugi pa se on samo ponovi i dobijemo novu riječ *PO*, koju upisujemo u rječnik i indeksiramo (dodijelimo kod). Četvrti korak je malo drugačiji. Radna riječ je ovdje sljedeći simbol u nizu, odnosno *O*, a kada mu dodamo sljedeći simbol on postaje *OP*, što već postoji u rječniku pa OP postaje naša nova riječ. Kako imamo još simbola na ulazu, ponavljamo iteraciju algoritma i dobijemo za *RadnaRiječ + NoviSimbol OPO*, što ne postoji u rječniku pa je bilježimo i indeksiramo te na izlaz kodera opet šaljemo kod. Ovog puta to je kod (indeks) dodijeljen riječi *OP*, a to je (4). U petom koraku ponavlja se sličan scenarij pa dobijemo novu riječ *OPOR*, budući da nam je *OPO* već poznato i postoji u rječniku. I na kraju nam ostaje zadnji korak algoritma u kojemu kodiramo posljednji simbol i na izlaz kodera šaljemo njegov kod, a to je (3) pošto je posljednji simbol *R*. Kako više nema simbola, kodiranje je završeno, a na izlazu kodera imamo sljedeće kodove: (1) (2) (2) (4) (7) (3).

Dekoderu je poznat početni rječnik, koji je postojao pri kodiranju, ali nije poznat prošireni rječnik koji je nastao kodiranjem te se isti mora napraviti i prilikom dekodiranja. Samo dekodiranje se provodi na sljedeći način. U prvom koraku na dekoder dolazi kod (1). Budući da je dekoderu poznat originalni rječnik on ovo dekodira kao simbol *O*. Radna riječ u ovom koraku je *O*, a pošto ona već postoji u rječniku nema njegovog proširivanja. U drugom korak na dekoder dolazi kod (2) pa se iz rječnika dekoder očitava simbol *P*. Radna riječ sada postaje *P*. Proširena riječ *OP* dobiva se tako da se s lijeva doda prethodni simbol iz poruke novom dekodiranom simbolu. Ova riječ ne postoji u rječniku te se dodaje i indeksira. U trećem koraku na dekoder dolazi kod (2) koji se dekodira kao P. Dodajući prethodni simbol dobijemo *PP* koji ne postoji u rječniku te se dodaje i indeksira. Nova radna riječ je *P*. U četvrtom koraku na dekoder dolazi kod (4) koji se pomoću rječnika dekodira kao *OP*. Pošto je ovo riječ od dva simbola rade se dvije stvari. Uzima se radna riječ iz prošlog koraka, a to je *P* te se na nju, s desne strane, dodaje prvi

simbol dekodirane riječi, a to je *O*, i dobije se *PO*. Ona ne postoji u rječniku pa se dodaje i indeksira. Nova radna riječ sada postaje *O*. Zatim se isto ponovi s drugim simbolom dekodirane riječi, a to je *P* koji se dodaje na novu radnu riječ *O* i dobije se *OP*. Ova riječ već postoji u rječniku te se ostavlja, a nova radna riječ je *OP*. U petom koraku na dekoder dolazi kod (7) kojeg nema u rječniku pa ga naizgled trenutno ne možemo dekodirati. Ovaj problem ćemo riješiti istim postupkom kao u prethodnom koraku. Dakle iz prethodnog koraka znamo da nova riječ počinje s *OP* te za sada imamo riječ oblika *OPx*, gdje je *x* nepoznati simbol. Nepoznanicu *x* otkrivamo dodajući prethodnoj radnoj riječi *OP* simbol *O* te za novu riječ dobijemo *OPO*, koju upisujemo u rječnik i indeksiramo. Ostaje još šesti korak u kojemu na dekoder dolazi kod (3) i pomoću rječnika dobijemo simbol *R*. Time je postupak dekodiranja, koji je prikazan i sljedećom tablicom, završen.

KORAK	ULAZ DEKODERA	DEKODIRANI SIMBOLI	SADRŽAJ RJEČNIKA
1	(1)	O	
2	(2)	P	(4)OP
3	(2)	P	(5)PP
4	(4)	OP	(6)PO
5	(7)	OPO	(7)OPO
6	(3)	R	

Glavna prednost metoda rječnika je ta što su vrlo efikasne u kodiranju simbola s izvora s memorijom te što kodiranjem nizova simbola koji se ponavljaju značajno smanjuju poruku ili niz simbola koje kodiraju.

Metode koje su izložene u ovom radu zapravo su samo osnovne metode kodiranja i samim time najjednostavnije. Dugi niz godina kroz koje su one korištene donio je i postupni napredak u samim algoritmima kao i nove algoritme. Jedan od danas najpoznatijih standarda koji koristi metode rječnika za kodiranje je Zip standard, odnosno format kodiranja. Metode rječnika prvenstveno se primjenjuju za kodiranje teksta pa se tako i njihov budući napredak vidi u ovom smjeru. Novije inovacije sadrže metode koje koriste više rječnika, a koristi se onaj koji najviše odgovara kontekstu u kojem se kodiranje provodi. Neki „moćniji“ modeli u obzir uzimaju i

gramatičke i semantičke parametre, ali ipak još je u toku potraga za efikasnijim metodama koje bi kompresiju provodile s manje od dva bita po simbolu.[36] Modeli koji obuhvaćaju više rječnika odgovarali bi na primjer univerzalnom koderu koji bi u idealnoj situaciji mogao kodirati riječi iz svih svjetskih jezika.

[36] Jayant, N. (ur.): Signal Compression: Coding of Speech, Audio, Text, Image and Video. Singapore: World Scientific, 1997., str. 208.

7. METODE SKRAĆIVANJA NIZA

Ova metoda temelji se na činjenici da se u nekom nizu simbola koji kodiramo može dogoditi da se neki simbol ponavlja više puta uzastopno. Tada taj niz simbola možemo skraćeno zapisati, odnosno kodirati korištenjem ove metode. Npr. uzmimo sljedeći niz simbola:

345643256000000000000000000000,

u kojem se na kraju ponavlja niz od 20 nula. Ako sada isti niz napišemo na sljedeći način:

345643250f20,

onda smo taj niz skratili, odnosno kodirali kodom f20, gdje je f „zastavica“ koja označava početak niza nula, dok je u nastavku naveden broj nula. U općenitijem slučaju vjerojatno nećemo imati niz nula nego neki općeniti niz simbola poput:

ABCDEEEEEEEEEEEEEEEFGH,

koji također upotrebom metode skraćivanja niza možemo kodirati na sljedeći način:

ABCDE!16GH,

U ovom smo slučaju umjesto niza simbola E pisali kod E!16 koji nam govori da na ovom mjestu u poruci slijedi 16 simbola E. Dakle, vrlo je jednostavno, uz pomoć metode kodiranja nula zastavicom f i metode kodiranja uzastopnih simbola pomoću zastavice !, skratiti neki niz na samo nekoliko simbola. Ali hoćemo li ovom metodom kodirati recimo sljedeći niz:

ABBBCD.

Odgovor je naravno ne, zato što bi pisanje B!3 zauzelo isti broj mjesta kao i broj simbola. Ovo je ujedno i donja granica upotrebljivosti ove metode. Ako niz simbola zauzima isti ili manji broj

mjesta od koda koji bi koristili za njihovo kodiranje onda je takvo skraćivanje nepotrebno i zapravo suvišno jer se njime ne postiže sažimanje poruke. Gornje granice korisnosti zapravo i nema pošto se bilo koji niz od bilo kojeg broja simbola koji se ponavljaju, može skratiti. Što je takav niz veći, ova metoda je efikasnija.

Metoda, u kojoj se niz simbola koji se uzastopno ponavljaju zamjenjuje zastavicom, nazivamo slijednim kodiranjem. Algoritam pomoću php programskog jezik u ovom slučaju glasi:[37]

```
<?php
function encode($str) {
  return preg_replace ('/(.)\1*/e', 'strlen($0) . $1', $str);
}

function decode($str) {
  return preg_replace ('/(\d+)(\D)/e', 'str_repeat($2, $1)', $str);
}

echo
encode('WWWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWWW'),
"\n";
echo decode('12W1B12W3B24W1B14W'), "\n";
?>
```

Ovaj kod niz simbola funkcijom encode, pri čemu je taj niz zapravo neki string, zamjenjuje s oznakom duljine stringa i zastavicom. Funkcija decode zamjenjuje output funkcije encode s originalnim nizom.

Osnovni nedostatak ove metode je taj što mora postojati niz simbola koji bi se mogao skratiti, odnosno baš je nužno da istovrsni simboli budu nanizani jedna do drugoga. Same prednosti su očite i pri dugim nizovima mogu biti i više nego značajne, dok se sama metoda upravo zbog navedenog nedostatka najčešće koristi kao dio drugih metoda, odnosno istovremeno s drugim metodama, radi postizanja bolje kompresije.

[37] RosettaCode.org: Run-Length Encoding. URL: http://rosettacode.org/wiki/Run-length_encoding#PHP (07.09.2016.)

8. ZAKLJUČAK

Kodiranje je danas vrlo raširen postupak. Ono se najčešće koristi kako bi se smanjila veličina određene datoteke, poput slike, videa, glazbe i sl. Samim time smanjuje se i količina memorije potrebne za njegovu pohranu, kao i vrijeme prijenosa i procesiranja. To se postiže npr. zamjenjivanjem više istovrsnih originalnih elementa s jednim ili manjim brojem elemenata kodne riječi. U situacijama koje zahtijevaju kompresiju bez gubitka, jer je bitno očuvati izvornu kvalitetu slike ili originalne poruke (npr. različite medicinske i satelitske snimke), nužnim se pokazuju metode entropijskog kodiranja. One provode kompresiju bez gubitka i, što je vrlo bitno, sažimanje s visokim omjerima kompresije, što ih čini idealnim za ovakve zadatke. Kompresiju bez gubitka vrše kreiranjem optimalnog koda koji zahtjeva da prosječna duljina kodne riječi poruke bude unutar 1 bita entropije. U okviru entropijskog kodiranja se načelno simboli s većom vjerojatnosti pojavljivanja kodiraju s kraćim kodnim riječima, a simboli s manjom vjerojatnosti pojavljivanja kodnim riječima veće duljine. Simboli predstavljaju elemente objekta koji se kodira. Kod teksta su to slova, a kod slike to mogu biti pikseli.

U ovom je radu prezentirano pet metoda entropijskog kodiranja. Kada ćemo koju od njih primijeniti ovisi o različitim čimbenicima. Npr. metode skraćivanja niza koriste se kada imamo poruku u kojoj se neki element uzastopno ponavlja veliki broj puta. S druge strane, metode rječnika su najkorisnije za kodiranje teksta. Huffmannovo i aritmetičko kodiranje mogu se upotrijebiti u različitim situacijama. Vrlo su efikasne za primjenu u slučajevima kodiranja slike, pri čemu je kompresija postignuta primjenom aritmetičkog kodiranja ipak nešto veća. No, prednost je Huffmannovog kodiranja u jednostavnosti postupka. Također postoje situacije u kojima je najbolje kombinirati dvije ili više metoda kodiranja.

Na kraju se može zaključiti da je primjena kodiranja neizbježna u današnjem svijetu u kojem dominiraju informacijske i komunikacijske tehnologije. Njima se ne samo skraćuje vrijeme i brzina slanja i primanja kao i prostor za pohranu, već se reduciraju i troškovi prijenosa i pohranjivanja.

9. LITERATURA

- Bing, B.: Next-Generation Video Coding and Streaming. Hoboken: Wiley, 2015.
- Bosi, M., Goldberg, R.E.: Introduction to Digital Audio Coding and Standards. New York: Springer Sciance + Business Media, LLC, 2003.
- Gibson, J.D., Berger, T., Lookabaugh, T., Lindbergh, D., Baker, R.L.: Digital Compression for Multimedia: Principles and Standards. Amsterdam: Morgan Kauffman Publishers, 1998.
- Jayant, N. (ur.): Signal Compression: Coding of Speech, Audio, Text, Image and Video. Singapore: World Scientific, 1997.
- Jones, G.A., Jones, J.M.: Information and Coding Theory. London: Springer, 2000.
- Kao, M.-Y. (ur): Encyclopedia of Algorithms. New York: Springer, 2008.
- Leondes, C.T. (ur.): Database and Data Communication Network Systems:Techniques and Applications, Volume 1. Amsterdam: Academic Press., 2002.
- Mandal, M.Kr.: Multimedia Signals and Systems. Boston: Kluwer Academic Publishers, 2003.
- Müller, P.: Entropy Coding. URL: http://www.icsy.de/studium/vorlesung/ws0910_MMS/pdf/05-compression-02.pdf (07.09.2016.)
- Muniswamy, V.V.: Design And Analysis Of Algorithms. New Delhi: I.K. International Publishing House Pvt. Ltd, 2009.
- Pandžić, I.S., Bažant, A., Ilić, Ž., Vrdoljak, Z., Kos, M., Sinković, V.: Uvod u teoriju informacije i kodiranje. Zagreb: Element d.o.o., 2007.
- Pauše, Ž.: Uvod u teoriju informacije, treće izdanje. Zagreb: Školska knjiga, 2003.
- RosettaCode.org: Huffman Coding. URL: http://rosettacode.org/wiki/Huffman_coding (07.09.2016.)
- RosettaCode.org: Run-Length Encoding. URL: http://rosettacode.org/wiki/Run-length_encoding#PHP (07.09.2016.)

- Salomon, D.: Data Compression: The Complete Reference, Fourth Edition. London: Springer, 2007.
- Sayood, K. (ur): Lossless Compression Handbook. Amsterdam: Academic Press, 2003.
- Shi, Q.J., Sun, H.: Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms and Standards. Boca Raton: CRC Press, 2008.
- Sinković, V.: Informacija, simbolika i semantika: načela i primjena teorije informacije. Zagreb: Školska knjiga, 1997.
- Wikipedia: Abraham Lempel. URL: https://en.wikipedia.org/wiki/Abraham_Lempel (07.09.2016.)
- Wikipedia: Arithmetic coding. URL: http://en.wikipedia.org/wiki/Arithmetic_coding (07.09.2016.)
- Wikipedia: Claude Shannon. URL: https://en.wikipedia.org/wiki/Claude_Shannon (07.09.2016.)
- Wikipedia: David A. Huffman. URL: https://en.wikipedia.org/wiki/David_A._Huffman (07.09.2016.)
- Wikipedia: Jacob Ziv. URL: https://en.wikipedia.org/wiki/Jacob_Ziv (07.09.2016.)
- Wikipedia: Jorma Rissanen URL: https://en.wikipedia.org/wiki/Jorma_Rissanen (07.09.2016.)
- Wikipedia: Robert Fano. URL: https://en.wikipedia.org/wiki/Robert_Fano (07.09.2016.)
- Wikipedia: Terry Welch. URL: https://en.wikipedia.org/wiki/Terry_Welch (07.09.2016.)
- Wu, H.R., Rao, K.R. (ur.): Digital Video Image Quality and Perceptual Coding. Boca Raton: CRC Press, 2006.
- You, Y.: Audio Coding: Theory and Applications. New York: Springer, 2010.
- Youtube.com: Claude Shannon - Father of the Information Age. URL: https://www.youtube.com/watch?v=z2Whj_nL-x8 (07.09.2016.)

ŽIVOTOPIS

Zovem se Mario Ožuška. Rođen sam 27. rujna 1987. godine u Osijeku, gdje sam završio Osnovnu školu Ljudevita Gaja. Nakon toga upisao sam srednju Ekonomsku i upravnu školu, također u Osijeku. Po završetku Preddiplomskog studija Fizike na Odjelu za fiziku Sveučilišta Josipa Jurja Strossmayera u Osijeku, akademske 2009./2010. godine upisao sam Diplomski studij Fizike i informatike. Radio sam kao nastavnik informatike u Osnovnoj školi Ljudevita Gaja u Osijeku i nakon toga kao profesor informatike u 1. Gimnaziji, također u Osijeku.