

Jednodimenzionalna Facsimil kompresija

Seminarski rad

Nikola Zlatkov 16593

Stefan Đurić 16574

Sadržaj

[Pojam kompresije 2](#_Toc63018224)

[Upotreba kompresije 2](#_Toc63018225)

[Slike 2](#_Toc63018226)

[Audio 3](#_Toc63018227)

[Video 4](#_Toc63018228)

[Uvod u facsimil kompresiju 5](#_Toc63018229)

[Huffman kodiranje 6](#_Toc63018230)

[Run-length encoding 7](#_Toc63018231)

[Jednodimenzionalno facsimil kodiranje 8](#_Toc63018232)

[Implementacija 12](#_Toc63018233)

[FNode.cs 12](#_Toc63018234)

[Code.cs 12](#_Toc63018235)

[Facsimile.cs 12](#_Toc63018236)

[Primena 13](#_Toc63018237)

[Pokretanje 14](#_Toc63018238)

[Literatura 15](#_Toc63018239)

Link od projekta na GitHub-u: (<https://github.com/zlatkovnik/Fascimile_Compression>)

# Pojam kompresije

U obradi signala, kompresija podataka je proces kodiranja informacija koristeći manje bitova od originalne veličine. Bilo koja kompresija je sa (***lossy***) ili bez (***lossless***) gubitaka. Kompresija bez gubitaka smanjuje bitove identifikovanjem i uklanjanjem statističke suvišnosti. Nijedna informacija se ne gubi kompresijom bez gubitaka. Kompresija s gubicima smanjuje bitove uklanjanjem nepotrebnih ili manje važnih informacija. Uobičajeno je da se uređaj koji vrši kompresiju podataka naziva enkoder, a onaj koji vrši preokret procesa (dekompresiju) kao dekoder.

Proces smanjenja veličine datoteke podataka često se naziva kompresijom podataka. Kod prenosa podataka to se naziva izvorno kodiranje (***source coding***), odnosno kodiranje izvršeno na izvornom uređaju pre nego što se podaci sačuvaju ili prenesu.

Kompresija je korisna jer smanjuje resurse potrebne za čuvanje i prenos podataka, ali se troše računarski resursi u procesima kompresije i dekompresije. Kompresija podataka je predmet kompromisa složenosti prostora i vremena. Na primer, kompresija za video format može zahtevati skup hardver da bi se snimak dovoljno brzo dekompresovao dok se gleda, a opcija da se video u potpunosti dekompresuje pre gledanja može biti nezgodna ili zahtevati dodatno skladištenje. Metoda kompresije podataka uključuje kompromise između različitih faktora, uključujući stepen kompresije, količinu izobličenja koja se uvodi (kada se koristi kompresija podataka sa gubicima) i računarske resurse potrebne za kompresiju i dekompresiju podataka.

# Upotreba kompresije

### Slike

Kompresija slike je vrsta kompresije podataka koja se primenjuje na digitalne slike radi smanjenja troškova za skladištenje ili prenos. Algoritmi mogu koristiti vizuelnu percepciju čoveka i statistička svojstva slike da bi pružili superiorne rezultate u poređenju sa generičkim metodama kompresije podataka koje se koriste za druge digitalne podatke.

Kompresija slike može biti sa gubicima ili bez. Kompresija bez gubitaka je poželjna u arhivske svrhe, a često i za medicinske snimke, tehničke crteže, grafičke radove ili stripove. Metode kompresije sa gubitkom su korisne kada se koriste pri malim brzinama prenosa, ali uvode artifakte kompresije. Metode sa gubicima su posebno pogodne za prirodne slike kao što su fotografije, u aplikacijama gde je manji gubitak kvaliteta prihvatljiv da bi se postiglo značajno smanjenje brzine prenosa podataka. Za kompresiju sa gubicima koja stvara zanemarljive razlike možemo reći da je vizuelno bez gubitaka.

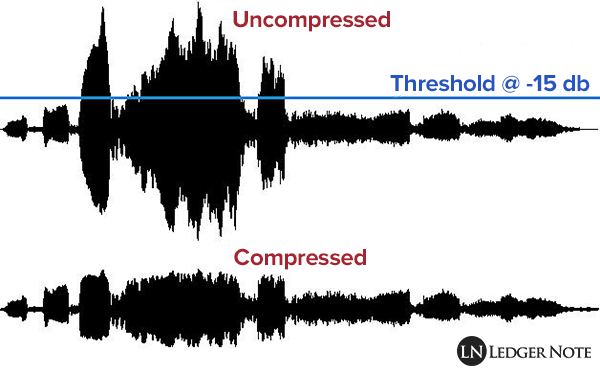


[Originalna slika | kopresovana slika]

### Audio

Kompresija audio podataka može smanjiti količinu podataka za prenos i zahteve za skladištenjem audio podataka. Algoritmi kompresije zvuka implementirani su softverski kao audio kodeci. I kod kompresije sa gubicima i bez gubitaka, redukovanost informacija se smanjuje, koristeći metode kao što su kodiranje, diskretna kosinusna transformacija kvantizacije i linearno predviđanje da bi se smanjila količina informacija koja se koristi za predstavljanje nekompresovanih podataka.

Kompresija se može koristiti za suptilnu korekciju numere kako bi je učinila prirodnijom i razumljivijom bez dodavanja izobličenja, čiji je rezulatat pesma koju je „ugodnije“ slušati. Pored toga, mnogi kompresori, bilo hardverski i softverski, imaće jedinstven zvuk koji se može koristiti za dodavanje divnih boja i tonova u inače beživotne numere. Prekompresovanje muzike je može previše izobličiti, tako da ona izgubi originalnu zamisao.

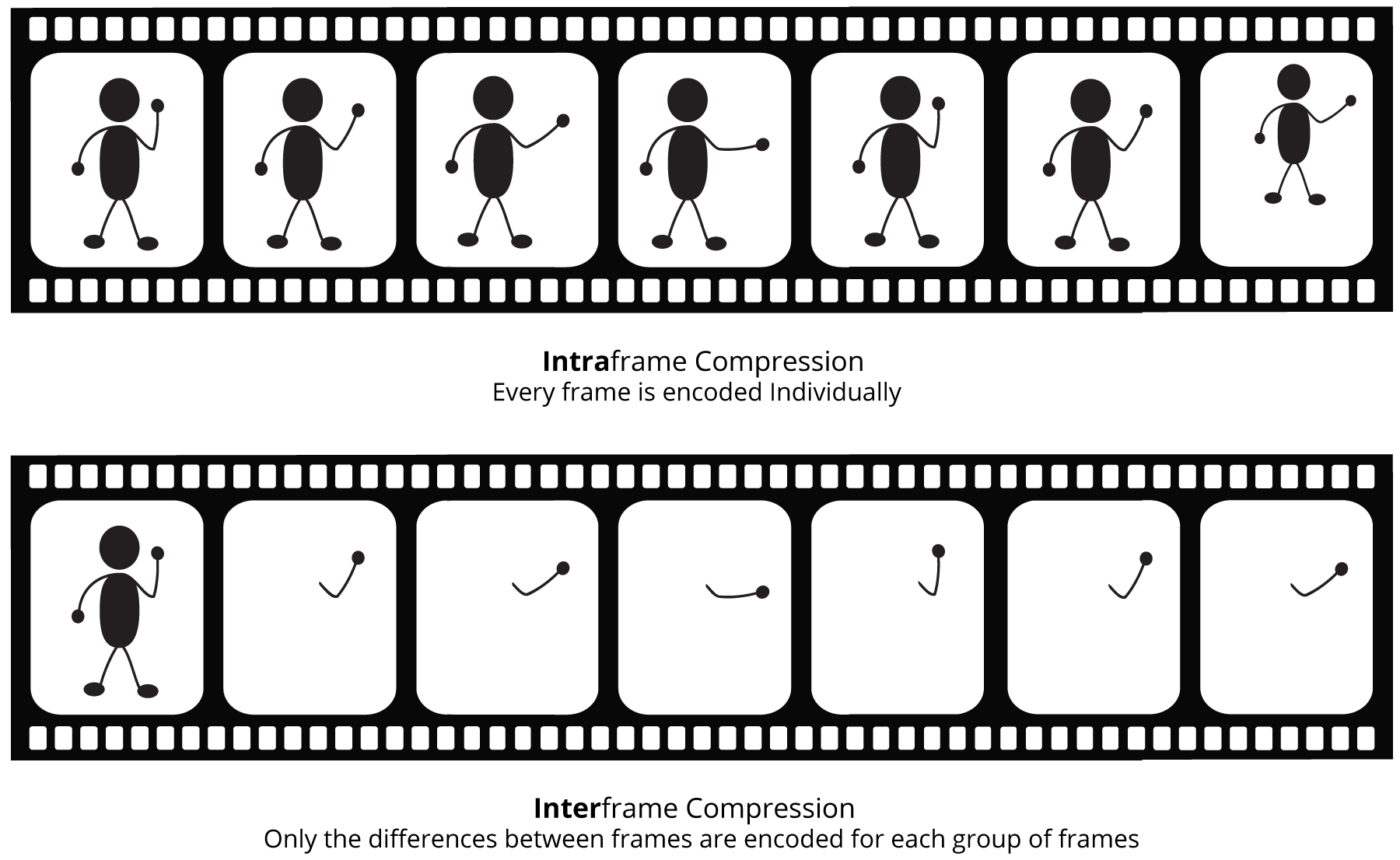


### Video

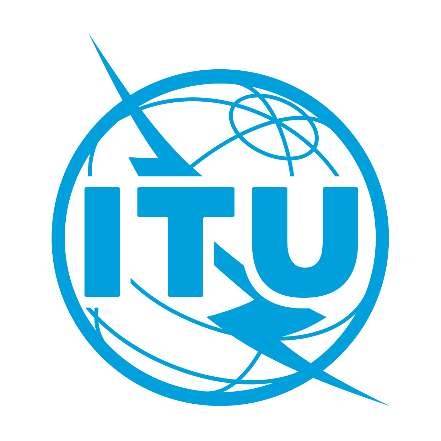
U praksi se većina video kodeka koristi zajedno sa tehnikama kompresije zvuka za čuvanje odvojenih, ali komplementarnih tokova podataka kao jedan kombinovani paket koristeći takozvane formate kontejnera (***container formats***).

Video podaci mogu biti predstavljeni kao niz smena nepokretnih slika. Takvi podaci obično sadrže ogromne količine redundantnih podataka. Algoritmi video kompresije pokušavaju da smanje redundantnost i kompaktnije čuvaju informacije.

Većina formata i kodeka za kompresiju video zapisa koriste prostornu i vremensku redundantost (npr. kroz kodiranje razlika između susednih frejmova). Sličnosti se mogu kodirati samo čuvanjem razlika između npr. tehnikama: susedni frejmovi (***inter-frame***) ili prostorno susedni pikseli (***intra-frame***).



# Uvod u facsimil kompresiju

Kompresija podataka je posebno važna kada se slike prenose preko zajednice komunikacione linije, jer korisnik obično čeka na prijemniku i željan da nešto vidi brzo. Dokumenti koji se prenose između faks mašina šalju se kao bitmape, zbog toga je bila potrebna standardna metoda kompresije podataka kada su te mašine postale popularne. Nekoliko metode je razvio i predložio ***ITU-T***.

***ITU-T*** je jedan od četiri celine ***Inernational Telecommunications Union*** (***ITU***), sa sedištem u Ženevi, Švajcarska (<http://vvv.itu.ch/>). Izdaje preporuke standarda koji se primenjuju na modeme, V.24 konektori, i sl. Iako nema ovlašćenje za sprovođenje, standardi koje preporučuje su opšte prihvaćeni i usvojeni u industriji. Do marta 1993. ITU-T je bio poznat kao Savetodavni komitet za međunarodne telefone i telegrafe. (Comit ́e Consultatif International T ́el ́egraphique et T ́el ́ephonique).

Prvi standardi kompresije podataka koje je razvio ITU-T bili su T2 (takođe poznati kao Grupa 1) i T3 (Grupa 2). Oni su sada zastareli i zamenjeni su T4 (Grupa 3) i T6 (Grupa 4). Grupu 3 trenutno koriste sve faks mašine dizajnirane da rade sa javnom telefonskom mrežom (***PSTN***). Ovo su mašine imamo kod kuće, a oko 2000. godine oni rade na maksimalnim brzinama od 9.600 baud. Grupu 4 koriste faks mašine dizajnirane za rad na digitalnoj mreži, kao npr ISDN. Imaju tipične brzine od 64K baud. Obe metode mogu stvoriti kompresiju odnosa 10: 1 ili bolje, smanjujući vreme prenosa tipične stranice na oko jedan minut kod prvog, a nekoliko sekundi kod drugog. Grupa 3 koristi kombinaciju ***Run length encoding*** (***RLE***) i ***Huffman coding***.

# Huffman kodiranje

U računarskim naukama i informatici ***Huffmanov kod*** je poznat kao vrsta optimalnog prefiks kodiranja koja se obično koristi za kompresiju podataka bez gubitaka. Proces pronalaženja ili upotrebe takvog koda odvija se pomoću ***Huffmanovog kodiranja***, algoritma koji je razvio ***David A. Huffman*** dok je studirao za doktorat na MIT-u i objavljen u radu iz 1952. godine „Metoda za izgradnju kodova sa minimalnim redundantnošću“.

Izlaz iz Huffmanovog algoritma može se posmatrati kao tabela kodova promenljive dužine za kodiranje izvornog simbola. Algoritam izvodi ovu tabelu iz procenjene verovatnoće ili učestalosti pojavljivanja za svaku moguću vrednost izvornog simbola. Kao i kod drugih metoda kodiranja, uobičajeni simboli su uglavnom predstavljeni koristeći manje bitova od neuobičajenih simbola. Huffmanova metoda se može efikasno primeniti, pronalaženje koda je efikasno ako su kodovi sortirani po težini (frekvenciji). Međutim, iako je ovo kodiranje optimalno među metodama koje odvojeno kodiraju simbole, Huffmanovo kodiranje nije uvek optimalno među svim metodama kompresije, zamenjuje se aritmetičkim kodiranjem ili asimetričnim numeričkim sistemima u slučaju da je potreban bolji stepen kompresije.

|  |  |  |
| --- | --- | --- |
| Karakter | Frekvencija | Kod |
| space | 7 | 111 |
| a | 4 | 010 |
| e | 4 | 000 |
| f | 3 | 1101 |
| h | 2 | 1010 |
| i | 2 | 1000 |
| m | 2 | 0111 |
| n | 2 | 0010 |
| s | 2 | 101 |
| t | 2 | 0110 |
| l | 1 | 11001 |
| o | 1 | 00110 |

[Na ovoj tabeli Huffman koda možemo uočiti da je kod kraći što je učestalost u nekom dokumentu veća]

# Run-length encoding

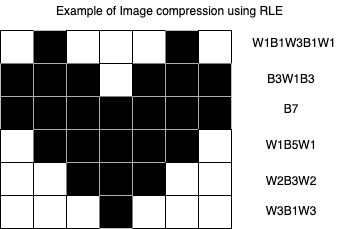
Kodiranje dužine ponavljanja (***RLE***) je oblik kompresije podataka bez gubitaka u kojem se niz podataka (sekvence u kojima se ista vrednost ponavlja u sekvencijalnim elementima) čuvaju kao jedna vrednost i broj ponavljaja, a ne kao originalno. Ovo je najkorisnije za podatke koji sadrže mnogo takvih ponavljanja. Kao na primer jednostavne grafičke slike poput ikona, crno belih dokumenata, linijskih crteža i animacija. Nije korisno sa datotekama koje nemaju puno ponavljanja, jer bi moglo u velikoj meri povećati veličinu datoteke.

Na primer, razmotrite dokument koji sadrži običan crni tekst na punoj beloj pozadini. U praznom prostoru biće mnogo dugih ponavljanja belih piksela, a unutar teksta mnogo kratkih crnih piksela. Hipotetička linija skeniranja, sa C koji predstavlja crni piksel, a B predstavlja belu, linija skeniranja dokumenta bi mogla da izgleda ovako:

BBBBBBBBBBBBCBBBBBBBBBBBBCCCBBBBBBBBBBBBBBBBBBBBBBBBC

Sa algoritmom kompresije podataka sa kodiranim protokom (RLE) primenjenim na gornju hipotetičku liniju skeniranja, može se prikazati na sledeći način: 12B1C12B3C24B1Cn. Ovo se može protumačiti kao niz od dvanaest belih tačaka, jedne crne, dvanaest belih, tri crne, itd.

Originalna linija skeniranja sadrži 67, a kompresovana staje u samo 18. Iako je stvarni format koji se koristi za čuvanje slika uglavnom binarni, a ne ASCII, princip ostaje isti.



# Jednodimenzionalno facsimil kodiranje

Faks uređaj skenira dokument red po red pretvarajući svaki red u male crne i bele tačke zvane ***pels*** (**P**icture **EL**ement). Horizontalna rezolucija je uvek 8,05 pela po milimetru (oko 205 pelova po inču). Tako je linija skeniranja široka 8,5 inča pretvorena u 1728 pela. Standard T4, međutim, preporučuje skeniranje samo oko 8,2 inča, proizvodeći tako 1664 pela po liniji skeniranja (ovi brojevi, kao i oni u sledećem pasusu, sa tačnošću od ± 1%).

Vertikalna rezolucija je 3,85 linija skeniranja po milimetru (standardni režim) ili 7,7 linija po mm (fini režim). Mnoge faks mašine takođe imaju vrlo fin način rada, skeniranje 15,4 linije po mm. Tabela ispod pretpostavlja da je stranica visoka 10 inča (254 mm) i prikazuje ukupan broj pelova po stranici i tipična vremena prenosa za tri načina bez kompresije. Vremena su duga, što pokazuje koliko je važna kompresija podataka u prenosu između faks mašina.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Linije skeniranja | Pels po liniji | Pels po stranici | Vreme (s) | Vreme (min) |
| 978 | 1664 | 1.670M | 170 | 2.82 |
| 1956 | 1664 | 3.255M | 339 | 5.65 |
| 3912 | 1664 | 6.510M | 678 | 11.3 |
| Deset inča je 254 mm. Broj pelova je u milionima, a vremena prenosa, na 9600 baud bez kompresije, su između 3 i 11 minuta, u zavisnosti od režima. Međutim, ako je stranica kraća od 10 inča, ili ako je veći deo bele boje, faktor kompresije može biti 10: 1 ili bolje, što rezultira vremenom prenosa između 17 i 68 sekundi. | | | | |

Da bi napravio kod grupe 3, ITU-T je izbrojao sve dužine ponavljanja bele i crne pels u osam dokumenata za „obuku“ za koje su smatrali da predstavljaju tipičan tekst i slike poslane faks mašinom i upotrebljenim Huffmanovim algoritmom u kome se svaka količina ponavljaja znaka posebno kodira.

Otkriveno je da su najčešće dužine ponavljaja 2, 3 i 4 crna piksela, pa su im dodeljeni najkraći kodovi što je prikazano na tabeli ispod. Slede dužine ponavljaja od 2–7 belih piksela, kojima su dodeljeni nešto duži kodovi. Većina dužina izvođenja bila je retka i dodeljivani su im dugi 12-bitni kodovi. Grupa 3 stoga koristi kombinaciju ***RLE*** i ***Huffman kodiranja***.

Interesantna stvar koju treba razmotriti da je dužina ponavljaja od 1664 belih pels kodirana kratkim kodom 011000 jer tipična faks mašina skenira linije koje su oko 8,2 inča širok (≈ 208 mm), tako da prazna linija skeniranja stvara 1.664 uzastopnih belih pelova. S obzirom da dužine ponavljaja mogu biti velike, Huffmanov algoritam je modifikovan. Kodovovima su dodeljene dužine ponavljaja od 1 do 63 pela i da se ponavljaju na dužine od 64 pela. Grupa 3 je prema tome modifikovani Huffman-ov kod (takođe nazvan ***MH***).

Kod dužine izvođenja je ili jedan završni kod (ako je dužina izvođenja kratka) ili jedan ili više izvedenih kodova, a zatim jedan prekidni kod (ako je dugačak). Evo nekoliko primera:

1. Dužina ponavljaja od 12 belih pela kodirana je kao 001000.
2. Dužina ponavljaja od 76 belih pela (= 64 + 12) kodira se kao 11011|001000 (bez “|”).
3. Dužina ponavljaja od 140 belih pela (= 128 + 12) kodirana je kao 10010|001000.
4. Dužina ponavljaja od 64 crne pela (= 64 + 0) kodira se kao 0000001111|0000110111.
5. Dužina ponavljaja od 2561 crnih pela (2560 + 1) kodirana je kao 000000011111|010.

Primetite da ne postoje ponavljaji dužine nula, ali su kodovi dodeljeni ponavljajima od nula crnih i nula belih pela. Ovi kodovi su potrebni za slučajeve kao što je u primeru pod rednim brojem 4, gde je dužina ponavljaja 64, 128 ili bilo koja dužina za koju je korišćen modifikovan kod. Još jedna suptilna karakteristika je da se kodovi kreću do 2561 pela, ali 8,5 inča široka linija skeniranja se sastoji od 1728 pelova. Objašnjenje je da možda postoje faks mašine (sada ili u budućnosti) napravljene za širi papir, tako da je kod Grupe 3 dizajniran da ih podrži u budućnosti.

Svaka linija skeniranja kodirana je odvojeno, a njen kod se završava posebnim 12-bitnim ***EOL*** kodom (kraj linije skeniranja) 000000000001. Svaka linija skeniranja takođe ima jednu belu pel koja joj je dodata s leve strane kod skeniranja. Ovo se radi kako bi se uklonila bilo kakva dvosmislenost kada se linija dekodira kod prijema. Nakon čitanja EOL-a za prethodni red, prijemnik pretpostavlja da nova linija počinje nizom belih pelova i ignoriše prvu od njih. Primeri:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

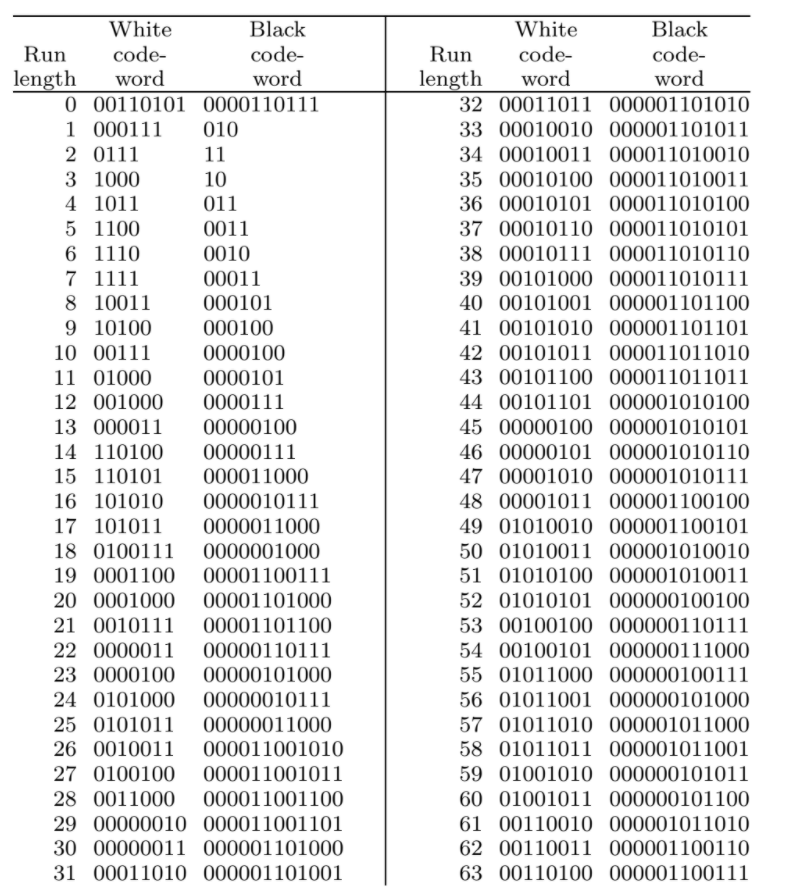
1. Linija od 14 pela kodirana je kao dužina ponavljanja 1B 3C 2B 2C 7B EOL, koji postaju 000111|10|0111|11|1111| 000000000001. Dekoder ignoriše jednostruke bele pelove na početku.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

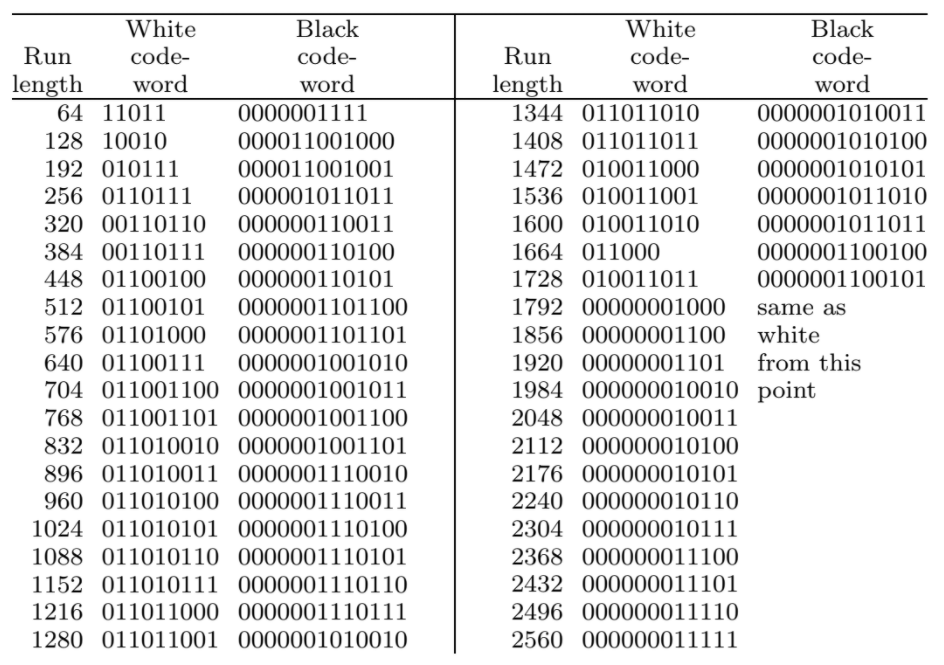
1. Linija je kodirana kao dužina ponavljanja 3B 5C 5B 2C EOL, koji postaje binarni niz 1000|0011|1100|11|000000000001.

Kod grupe 3 nema ispravku grešaka, ali se mogu otkriti mnoge greške. Jer je priroda Huffmanovog koda, da čak i jedan loš bit u prenosu može prouzrokovati prijemnik da ispadne iz sinhronizacije i da proizvede niz pogrešnih pelova. Ovo je razlog zašto je svaka linija skeniranja posebno kodirana. Ako prijemnik otkrije grešku, preskače bitove i traži EOL. Na ovaj način jedna greška može dovesti do primanja najviše jedne linije skeniranja netačno. Ako prijemnik ne vidi EOL nakon određenog broja linija, pretpostavlja se velika stopa grešaka i prekida se proces, uključujući obaveštavanje predajnika. Pošto su kodovi dužine između 2 i 12 bita, prijemnik otkriva grešku ako ne može da dekodira važeći kod nakon čitanja 12 bita.

Svakoj stranici kodiranog dokumenta prethodi jedan EOL, a zatim šest EOL kodova. Budući da je svaka linija kodirana odvojeno, ovaj metod je jednodimenzionalno kodiranje. Stepen kompresije zavisi od slike. Slike sa velikim susednim crnim ili belim područjem (tekst ili crno-bele slike) mogu biti jako dobro kompresovane. Slike sa mnogo kratkih ponavljaja poenakd mogu proizvesti negativnu kompresiju. Ovo se najčešće dešava u slučaju slika s nijansama sive (kao što su skenirane fotografije). Takve nijanse se proizvode polutoniranjem, koje pokriva područja sa mnogo naizmeničnih crnih i bele pela (broj ponavljaja dužine jedan).



[Tabela terminalnih kodova]



[Tabela modifikovanih kodova]

Za bolje razumevanje, zamislite liniju skeniranja gde sva ponavljanja imaju dužinu jedan (strogo naizmenični pelovi). Lako je videti da ovaj slučaj rezultira proširenjem. Kod dužine ponavljaja jednog belog pela je 000111, a kod jednog crnog pela je 010. Dva uzastopna pela različitih boja tako se kodiraju u 9 bitova. Pošto je za nekodirane podatke potrebno samo dva bita (01 ili 10), odnos kompresije je 9/2 = 4.5 (kompresovani podatak je 4,5 puta duži od nekompresovanog, „***veliko proširenje***“).

Standard T4 takođe omogućava umetanje bitova za dopunu između bitova podataka i EOL. To se radi u slučajevima kada je neophodna pauza ili zato što ukupan broj bitova koji se prenose za liniju skeniranja moraju biti deljivi sa 8. Bitovi za popunjavanje su nule.

Na primer:

Binarni niz 000111|10|0111|11|1111|000000000001 postaje 000111|10|0111|11|1111|00|0000000001 nakon dodavanja dve nule kao bitova za popunjavanje, dovodeći ukupnu dužinu niza do 32 bita (= 8 × 4). Dekoder vidi dve nule ispune, a zatim jedanaest nula znaka EOL, praćen jednom jedinicom, pa zna da je naišao na ispunu praćenu znakom EOL.

# Implementacija

Ovde ćemo prikazati našu implementaciju facsimile kompresije, proći ćemo kroz klase i metode i objasniti kako rade. Izvorni kod implementacije:

<https://github.com/zlatkovnik/Fascimile_Compression>

### FNode.cs

Je čvor koji koristimo za Huffmanovo stablo, u njemu se pamti ***frekvencija***, ***code*** struktura koja sadrži boju i broj ponavljanja te boje u dokumentu i pokazivače na levi i desni čvor u slučaju da postoje.

Metod ***CompareTo*** upoređuje dva čvora i vraća pozitivan broj ukoliko je frekvencija prvog veća, vraća nulu ako su istih frekvencija, negativan broj ako je drugi veći.

### Code.cs

Sadrži boju (crno ili belo) pela i broj sekvencijalnih ponavljanja boje u dokumentu.

### Facsimile.cs

Ima metodu ***Compress*** koja iz fajla učitava nule i jedinice kao karaktere, a zatim funkcijom ***GetColorsFromString*** čita nulu ili jedincu kao belu ili crnu boju respektivno i upisuje ih u niz, ***GetCodedPaper*** kodira taj niz tako što sve crne/bele nizove pelova kodira brojem ponavljanja i njihovom bojom, nakon toga generiše stablo funkcijom ***GenerateTree*** zatim se stablo i kodiran dokument čuvaju u fajl metodom ***SaveToFile***.

Metoda ***Decompress*** čita dokument koji je kodiran i binarno stablo iz fajla metodom ***LoadFromFile***, zatim dekodiran dokument čuva u tekstualni fajl ***SavePaperToTextFile*** metodom bez gubitaka.

***GenerateTree*** generiše čvorove stabla na osnovu dokumenta kodiranog brojem ponavljanja pelova metodom ***GetNodesFromCodes,*** zatim kreira red sa prioritetom i ubacuje sve čvorove u njega. Iz reda izbacuje dva elementa koja se ponavljaju minimalan broj puta tj. sa najmanjim frekvencijama. Na osnovu njih kreira se **mixed** čvor kome je frekvencija zbir frekvencija izbačenih čvorova, levi potomak je izbačeni čvor sa manjom frekvencijom, a desni je izbačeni čvor sa većom frekvencijom. Koren stabla se setuje na taj **mixed** čvor i on se ubacuje u red.

***SavePaperToTextFile*** kodirani dokument dekodira korišćenjem stabla, a zatim dekodirani dokument upisuje u tekstualni fajl.

Primer: (nije kako naš program kodira, ali je simboličan) 5W3B2W->WWWWWBBBWW

***SaveToFile*** u binarni fajl čuva stablo koje je potrebno za kasnije dekodiranje metodom ***SaveTreeToFile***,a zatim čuva kodirani dokument koristeći ***SavePaperToFileOptimized***.

***SaveTreeToFile*** čuva binarno stablo u dokument metodom inorder gde se prvo zapisuje boja, a zatim dužina ponavljaja, a u slučaju da potomak ne postoji upisujemo “#”.

***SavePaperToFileOptimized*** u ovoj metodi Huffmanove kodove upisujemo bit po bit u fajl.

***LoadFromFile*** koristećimetodu ***ReadTreeFromFile*** čita stablo iz fajla ranije opisanom metodom, a ***ReadCodedPaperFromFileOptimized*** koristi stablo da bi dekodirao kodiran dokument iz fajla.

***ReadTreeFromFile*** se zove rekurzivno i učitava stablo iz fajla po inorder metodi koja se koristi u ***SaveTreeToFile***.

***ReadCodedPaperFromFileOptimized*** bit po bit čita Huffmanove kodove iz fajla koristeći malopre učitano stablo.

***GetNodesFromCodes*** u hash tabeli ubacuje novo generisane kodove, a u slučaju da je neki kodova već ubačen inkrementira njegov broj ponavljanja u dokumentu (frekvenciju). Time se dobija svaka kombinacija dužina ponavljanja neke boje.

***GetCodedPaper*** niz nula i jedinicapretvara u boju i broj ponavljanja.

***GetPath*** se zove rekurzivno i vraća Huffmanov kod nekog koda (odnosno boju i broja ponavljanja) koristeći stablo.

# Primena

Naša implementacija ne razlikuje redove, već kodira dokument kao jednu ogromnu scan liniju. Na kraju dekodiranog dokumenta može se desiti da ima dodatne bitove, zbog toga što bi kod trebalo da bude umnožak broja 8. A to se dešava zato što se dopisuju nule na kraj kodiranog dokumenta kako bi mu veličina bila ceo broj bajtova. To nije problem jer se zna koja veličina dokumenta i ti dodatni pelovi se odbacuju (odsecaju).

Naravno u zavisnosti od dokumenta menja se efikasnost kompresije, ona u nekim primerima kodira dokument u odnosu 10:1, a nekad je kompresija i negativna. Recimo za dokument: 000000011100000001000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111000011100001110000111

,gde ima dosta ponavljanja četiri nule i tri jedinice, fajl od 733 bajta kodira u 63 bajta, umanjenje originalnog fajla je čak 91.5%. Dok se dokument sa sledećim nizom belih i crnih pelova: 101100111000101111100011100011 od 31 bajt, kodira u 46 bajta, kodiran fajl je 48% veći od originalnog. Međutim ako je dokument duži, što je i realno za dokumente koji se šalju faks mašinom, i gde se niz pelova stalno ponavlja kompresija postaje dosta efikasnija. Ako prethoni niz pelova ponovim dvanaest puta dobijam dokument od 361 (nije tačno 12 puta veći jer fajl ima fiksni header deo i EOF karakter), čija kodirana varijanta zauzima 90 bajta, veličina kodiranog fajla je 75% manja od originala.

# Pokretanje

Kod pokretanja moraju se uneti sledeći argumenti u komandnoj liniji, u sledećem formatu:

{ input fajl } {odredišni fajl } {[c, d] u zavisnosti od toga da li želimo da vršimo kompresiju ili dekompresiju}

Ako se vrši kompresija prvi argument mora da bude tekstualni fajl u kome su beli pelovi upisani nulom, a crni jedinicom, i na kraju fajla stoji EOF simbol „2“. Ako se vrši dekompresija prvi argument mora da bude binarni fajl koji sadrži kodirani dokument sa stablom.

Drugi argument je tekstualni kod kompresije, a binarni kod dekompresije. On se automatski generiše u slučaju da ne postoji.

Treći argument ako se ne navede, po defaultu se vrši kompresija.

Primer pokretanja iz projekta:

#### dotnet run -- input.txt output.bin c

#### dotnet run -- input.bin output.txt d

# Literatura

* **Data Compression** 3rd Edition by **David Salomon**
* **CCITT Encoding** (<https://www.fileformat.info/mirror/egff/ch09_05.htm>)
* **Huffman Coding** (<https://en.wikipedia.org/wiki/Huffman_coding>)
* **Data Compression** (<https://en.wikipedia.org/wiki/Data_compression>)
* **Audio Compression** (<https://www.uaudio.com/blog/audio-compression-basics/>)
* **Run Length Encoding** (<https://en.wikipedia.org/wiki/Run-length_encoding>)