V – Kompresija i integrititet podataka

- Da bi smo smestili velike fajlove podataka u računar, potrebno je izvršiti sažimanje (kompresovanje) tih podataka, odnosno smanjiti fajl

Kompresija podataka je način da se ista informacija zapiše sa manje zauzetog prostora na disku.

Kompresija je proces sažimanja podataka odnosno pretvaranje podataka u oblik koji zauzima manje memorije

- Osnovni cilj da se poslata informacija smanji na što manju veličinu
- Šta se postiže - manje podataka za slanje i manje memorije za pamćenje – bitni preduslovi za slanje multimedijalnih podataka
- Dva faktora su važna za svaku kompresiju: **prenosivost** i **performanse**

Primeri:

1. Slika od 1024 x1024 pixela (24 bita za boju) – 3 MB tj. 7 min za slanje pri brzini od 64 kbit/s
2. Video slika (20 000 pixela po slici x 24b = 4 800 000 b) tako da je potrebno da se pošalje **14 648 MB za 1s (25 slika u sekundi)**

V – Kompresija podataka

1. **Kompresija bez gubitaka podataka** (“*lossless compression*”) – predstavlja sažimanje kod koga ne dolazi do gubitka podataka i kvaliteta komprimovane informacije.

- Najpoznatiji formati koji upražnjavaju ovaj način sažimanja su: **PNG**(za slike) i **FLAC**(za audio), **ARJ**, **ZIP**, **RAR**, **CAB** i drugi.

2. **Kompresija sa gubicima podataka** (“*lossy compression*”) - je način sažimanja podataka sa unapred prihvatljivim malim gubicima.

- Koristi se uglavnom kod multimedijalnih podataka (zvuk i video).
- Najpoznatiji formati su **JPG** (slike), **MP3** (audio) i **MPEG** (video).
- Metode sa gubicima zasnivaju se na modelima ljudske percepcije (više kompresuju one atribute slike koji manje doprinose ukupnom izgledu)
- One uzrokuju degradaciju slike u svakom koraku (svakim sledećim korakom kompresije/dekompresije slika se degradira), ali najčešće omogućuju daleko veći procenat kompresije nego metode bez gubitaka.

5.1- Kompresija bez gubitaka podataka

Kod kompresije bez gubitaka dobija se fajl koji će posle dekompresije biti identičan originalu

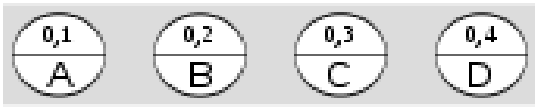
1. Frekventno zavisni kodovi
   - Hafmanovo kodiranje
   - Aritmetička kompresija
2. Run-Lenght kodiranje
   - Nizovi istog bita
   - Nizovi sa različitim karakterima
   - Faksimil kompresija
3. Lempel-Ziv (entropijsko kodovanje)
4. Relativno kodiranje
5. Kodovanje područja
6. PNG (*Portable Network Graphics*)

5.1.1 – Hafmanovo kodiranje

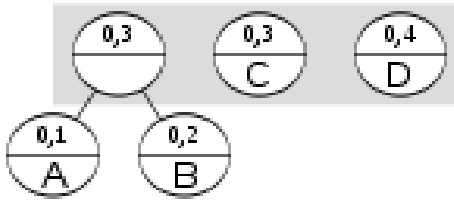
➢ ***Frekventno zavisni kodovi*** - znaci koji se češće pojavljuju, kodiraju se sa manjim brojem bitova

1. Svakom karakteru se dodeljuje binearno stablo koje se sastoji samo iz jednog čvora. Svakom stablu je dodeljena učestalost pojavljivanja karaktera koju nazivamo težina stabla.
2. Traže se dva najlakša stabla. Ako ih ima više biraju se bilo koja dva. Spajamo ih u jedno sa novim korenom, gde levo i desno podstablo odgovara ranijim stablima. Sumu spojenih stabala dodeljujemo kao sumu novom stablu.
3. Postupak se ponavlja sve dok ne dobijemo jedno stablo.
4. Kodiranje se vrši tako što se levoj grani dodeli bit „**0**“ a desnom bit „**1**“.

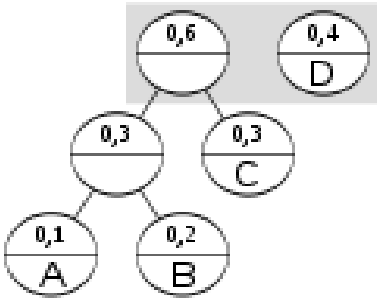
0)



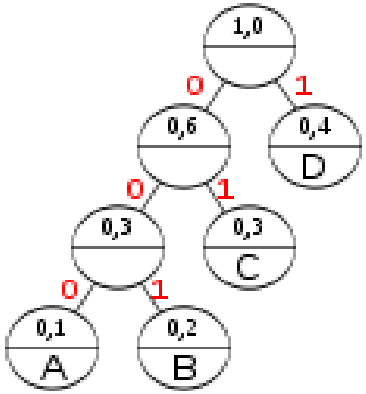
1)



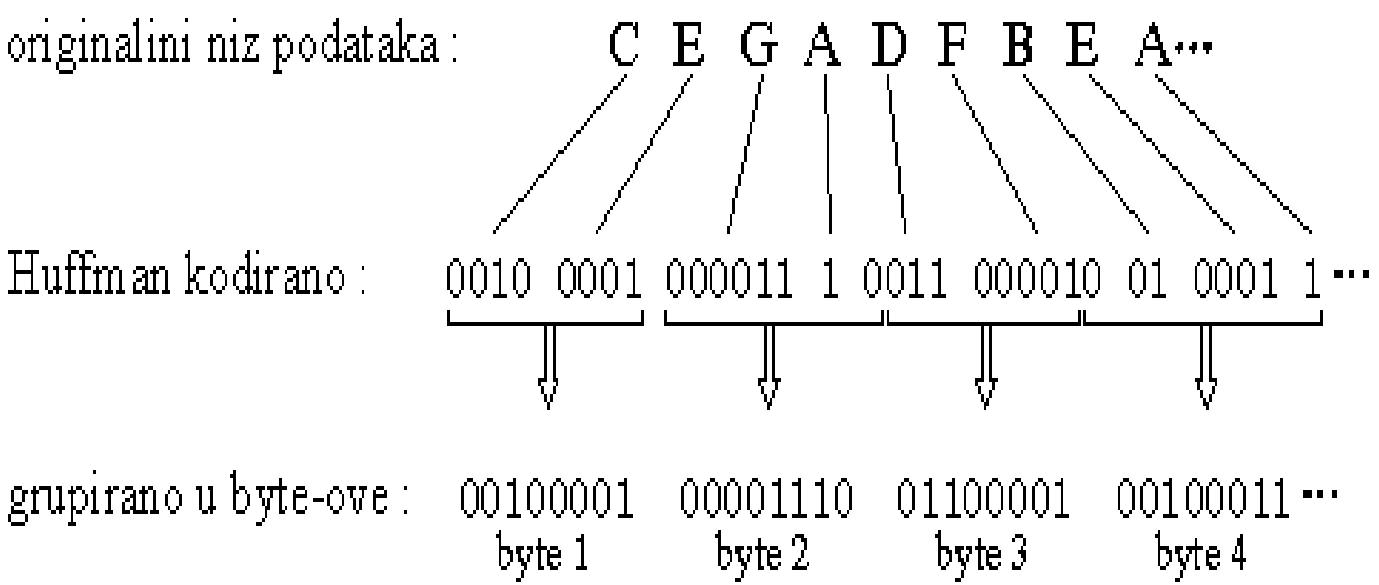
2)



3)



5.1.1 – Hafmanovo kodiranje



primjer kodne tablice :

slovo	vjerotojatnost	Huffman kod
A	.154	1
B	.110	01
C	.072	0010
D	.063	0011
E	.059	0001
F	.015	000010
G	.011	000011

Dva pravila moraju biti ispoštovana kod ovog kodiranja:

1. **No prefix** – nijedan kod slova na početku ne sme da sadrži kod drugog slova
2. Slova koja se češće ponavljaju (imaju veću težinu) moraju biti kodirana sa manjim ili isitim brojem bitova od slova koja se ređe ponavljaju

5.1.1 – Aritmetička kompresija

- Povorka podataka predstavlja se kao kao jedan realni broj.
- Sve 0 i 1 u okviru jedne povorke tretiraju se kao jedan jedini broj.
- Aritmetička kopresija funkcioniše tako što se uspostavlja jedna asocijacija između niza karaktera i realnog broja između 1 i 0
- Matematički gledano između 0 i 1 ima beskonačno mnogo brojeva
- Razlika u odnosu na Hafmanov kod je u tome što se opseg brojeva dodeljuje na osnovu učestanosti pojavljivanja pojedinih karaktera.

U odnosu na prethodni primer to bi bilo ovako:

A	25 %	(0, 0.25)
B	15 %	(0.25, 0.4)
C	10 %	(0.4, 0.5)
D	20 %	(0.5, 0.7)
E	30 %	(0.7, 1.0)

5.1.1 – Aritmetička kompresija

Algoritam za dodelu bi izgledao ovako:

1. Startujemo od intervala (x,y)=(0,1)
2. Pogledamo prvi karakter i utvrdimo odgovarajući podinterval (x,y) koji zavisi od učestanosti karaktera.
3. Redefiniše se interval (x,y) koji će sada predstavljati taj podinterval.
4. Ispitamo sledeći karakter i ponovo utvrdimo novi podinterval (x,y) u zavisnosti od učestalosti tog karaktera (na osnovu p i q). Ovo je identično kao i u koraku 2 ali sada radimo samo sa novim podintervalom koji je određen u koraku 2.
5. Koraci 3 i 4 se ponavljaju za svaki karakter.

Primer:

X(0,3) w = 0.9 - 0.3 = 0.6 Y(0,9)

p(0,25) i q(0,5)

x+**w** x **p**=0.3+0.6 x 0.25=0.45 **x**+**w** x **q**=0.3+0.6 x 0.5=0.6

5.1.1 – Aritmetička kompresija

- Obrnuti proces (dekriptovanje kod prijemnika) se sastoji u tome da se prvo utvrdi u kom delu intervala se nalazi primljeni broj.
- Tako se utvrđuje prvi karakter koji je poslat a zatim se na osnovu razlike određuju i ostali.
- Terminalni karakter – karakter do koga se vrši dekodiranje

Primer: *Primili smo broj **0.4067**. Odrediti niz koji smo primili ako je učestanost pojavljivanja karaktera kao što smo do sada radili:* A(0-0.25), B(0.25-0.4), C(0.4-0.5), D(0.5-0.7) i E(0.7-1).

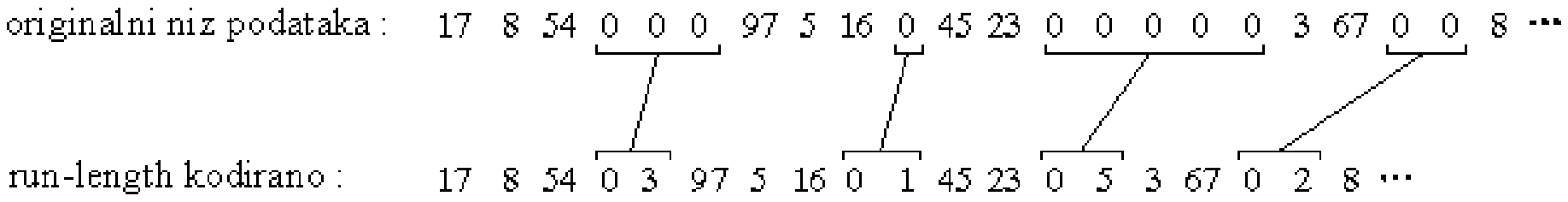
Obrnuti proces-dekodiranje:

Korak	***Interval***	***Širina***		***Karakter***	***N-p***	***(N-p)/širina***
1	0.4067	(0.4, 0.5)	0.1	**C**	0.0067	0.067
2	0.067	(0, 0.25)	0.25	**A**	0.067	0.268
3	0.268	(0.25,0.4)	0.15	**B**	0.018	0.12
4	0.12	(0, 0.25)	0.25	**A**	0.12	0.48
5	0.48	(0.4, 0.5)	0.1	**C**	0.08	0.8

Primili smo niz **CABAC**

5.1.2 – Run Lenght kodiranje

- Zasniva se na principu da se dugi niz bitova šalje kao jedan podatak.
- Koristi se činjenica da su u mnogim fajlovima česti nizovi istih vrednosti
- Veoma lako se primenjuje, bilo softverskim ili hardverskim alatima, kodiranje je jako brzo, kodirani kod se lako proverava i dekodiranje je veoma brzo i jednostavno



5.1.2 – Run Lenght kodiranje

1. Nizovi istog bita

Binarni niz: **0 …. 010……..0110……..……010……..…0110……..0** – 90 bitova

14 9 20 30 11

Kodirani niz: **1110 1001 0000 1111 0101 1111 1111 0000 0000 101** – 39 bitova

14 9 0 15 5 15 15 0 0 11

2. Nizovi sa različitim karakterima

HHHHHHHHHHKKKKKKKKKMMMMMMMMJJJ > 10H8K7M3J

3. Faksimil kompresija

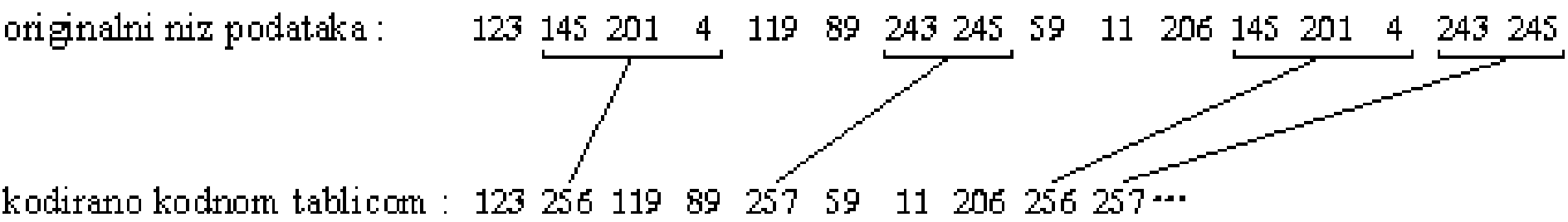
- Koristi se *modifikovani Hafmanov kod* - ti kodovi se daju za različite dužine niza belih i crnih tačaka i to od 0 do 63 i zatim samo za dužine od 64, 128, 192, 256 (umnožak od 64).
- Kodovi do dužine od 64 nazivaju se ***konačni kodovi*** a kodovi preko dužine 64 su ***kodovi doterivanja***.

5.1.3 – Entropijsko kodovanje (Lempel-Ziv)

- Traže se stringovi koji se najčešće ponavljaju i oni se kodiraju određenim kodom koji se sada upotrebljava umesto njih.
- Stepen kompresije 1 : 8 za prosečne GIF slike
- Problematična za implementaciju jer su nekada potrebne velike tabele

Primjer kodne tablice :

kodni broj	prijevod
0000	0
0001	1
⋮	⋮
0254	254
0255	255
0256	145 201 4
0257	243 245
⋮	⋮
4095	xxx xxx xxx



5.1 – Relativno i kodiranje područja

5.1.4 Relativno kodiranje

- Ovde se prenose samo razlike između signala a ne i ceo signal
- Znatno se smanjuju informacije koje se prenose (primer slike koja prenosi samo promene).

5.1.5 Kodovanje područja

- Poboljšana verzija *Run-Length* kodovanja koja koristi dvodimenzionalnu karakteristiku slika.
- Algoritam pokušava da pronađe pravougle regije jednakih karakteristika, koje se zatim kodiraju u opisnoj formi kao elementi s dve tačke i određenom strukturom.
- Performanse zavise od rešavanja vrlo kompleksnog problema pronalaženja najvećih područja jednakih karakteristika.
- Ovo je vrlo je efikasan način kodovanja, ali zbog svoje nelinearnosti onemogućuje hardversku implementaciju, te je relativno spor.

5.1.6 – Portable Network Graphics

- ***PNG*** je otvoreni grafički format namenjen kodiranju nepokretnih slika bez gubitaka, nastao kao zamena za GIF.
- PNG podržava slike zasnovane na paleti (24 bitnim RGB bojama - crvena, zelena i plava), *greyscale* slike (sivi tonovi) i RGB slike.
- PNG je razvijen kao patent i licencno je neopterećen
- Predstavlja poboljšanje u odnosu na GIF tehniku.
- Slika u *lossless* PNG datoteci može biti 5% -25% više komprimirana nego GIF datoteka iste slike.
- PNG se gradi na ideji o transparentnosti GIF slike i omogućuje kontrolu stepena transparentnosti, poznat kao **neprozirnost**.
- Pamćenje, restauriranje i ponovno pamćenje PNG slika neće degradirati kvalitet slike.
- PNG ne podržava animacije kao GIF.
- MNG (*Multiple Network Graphics*) - sličan PNG a namenjen animaciji

5.2 – Kompresija sa gubitcima

- Postoje neki podaci koji ne zahtevaju 100% podudarnost a da opet bude zadovoljena koegzistentnost informacije koja se prenosi.
- Obično je to slučaj sa podacima koji prenose multimedijalne informacije kao što su zvuk, slika ili video.
- Dva osnovna elementa koji utiču na sadržaj, veličinu i kvalitet slike su
  - broj piksela slike - odluku o broju piksela slike donosimo u zavisnosti od namene slike,
  - dubina piksela - informacija o boji svakog piksela slike čuva se u nizu bitova fiksne dužine.
- Performanse algoritama za kodovanje s gubicima se najčešće izražavaju preko dva faktora:

  1. faktor kompresije – objektivan faktor
  2. distorzija proizvedena nakon rekonstrukcije – zavisi od izbora slike

5.2 – Kompresija sa gubitcima

- U svim ovim metodama kompresije sa gubicima možemo prepoznati tri faze preko kojih se dolazi do kodirane informacije:

1. modelovanje slike (definicija transformacije koja se koristi) - usmeren je na korišćenje statističkih karakteristika slike (npr. korelacija). Pokušava se da što manji broj koeficijenata u transformisanom domenu sadrži što veći deo informacija originalne slike. Ova faza najčešće ne rezultira nikakvim gubitkom informacija.
2. kvantizacija parametara (kvantizacija podataka dobijenih transformacijom) - cilj kvantizacije je da smanji količinu podataka potrebnu za predstavljanje informacija u novom domenu. Kod kvantizacije u većini slučajeva dolazi do gubitka informacija.
3. kodovanje - optimizuje reprezentaciju informacija, te se može uneti detekcija grešaka.

5.2 – PCX i TIFF kompresija

5.2.1 PCX

- Jedan od najstarijih i najkorišćenijih bitmap formata (početkom 1980)
- PCX fajlovi mogu čuvati podatke o slikama dubine od 1, 4, 8 i 24 bita.
- Algoritam kompresije je RLE (*Run Length Encoding*).
- Što je niz duži, kompresija je veća.
- Najbolje rezultate - kompresija crno-bele grafike i jednostavnih crteža.
- Podržava ga većina skenera, fax programa i sistema stonog izdavaštva.

5.2.2 TIFF (*Tag(ged) Image File Format*)

- TIFF je razvijen sa ambicijom da postane standardni format za slike
- Jedan je od najšire podržanih fajl formata za čuvanje bitmapiranih slika
- TIFF je pravi izbor formata i ako sliku hoćemo da koristimo u nekom od programa za stono izdavaštvo.
- Pouzdaniji format od PCX-a, moćnije metode kompresije pa samim tim i manje veličine fajlova
- TIFF format podržava praktično sve dubine piksela i veliki broj metoda kompresije, pa se često dešava da ne bude uvek prepoznat u nekim programima.

5.2 – BMP i GIF kompresija

5.2.3 BMP (bitmap format)

- BMP je standardni format za bitmapiranu grafiku korišćen u Win OS.
- Pri kodiranju slike možemo se opredeliti za MS Windows ili OS/2
- Može se zadati dubina piksela a podržava dubine piksela 1, 4, 8 i 24 b.
- Podržava RLE algoritam kompresije za slike sa 4 ili 8 bita/pikselu.
- Čuvaju grafiku u formatu poznatom kao ***D**evice **I**ndepended **B**itmap*
- BMP fajlovi su obično bez kompresije pa zauzimaju dosta memorije
- Prednost : jednostavnost, visoko standardizovan i raširenost

5.2.4 GIF *(Graphics Interchange Format)*

- Koristi se za prikaz jednostavnih slika na WEB-u.
- Namenjen kompresiji slika koje ne zahtevaju veliki broj boja (do 256)
- Koristi se varijacija Lempel-Ziv kodiranja bez gubitaka.
- Komprimuje jednostavne slike sa velikim oblastima iste boje.
- GIF format je dobar izbor za crteže, crno-bele slike i za sitan tekst.
- Sve boje koje postoje na slici zadaju se u posebnoj tabeli boja (256 boja) **- paleti** ili CLUT(*Color Lookup Table*-tabela pretraživanja boja)
- Može se osigurati kompresija **3:1**, a uz neke dodatne operacije i 5:1.

5.2.5 – JPEG kompresija

- Kompresija slika s velikim brojem različitih boja – slike stvarnog sveta
- Dobar je za fotografije, ali nije baš uspešan pri kompresiji jednostavnih crtanih slika ili linija jer ima problema s oštrim rubovima.
- Napravljen je da koristi nesavršenosti ljudskog oka, odnosno činjenicu da se okom bolje primećuju male razlike u osvetljaju nego u boji.
- Moguće je imati kompromis između veličine slike i kvaliteta.
- Za dobar kvalitet slike se ne mogu mnogo kompresovati, ali ako nam nije jako važan kvalitet možemo postići visok stepen kompresije.
- Možemo birati između kvaliteta slike i brzine dekodovanja
- Jedini pravi nedostatak JPEG-a sastoji se od toga da svaki put kada kompresujemo/dekompresujemo sliku gubimo sve više informacija.
- Vrlo je važno ograničiti broj kompresija i dekompresija između početne i završne verzije slike.
- JPEG može osigurati kompresiju **20:1** sa svim bojama bez vidljivih gubitaka informacija (ako kvalitet nije bitan postiže se 100:1)
- Danas postoji i novi stanadard nazvan **JPEG-LS** koji omogućava veći nivo kompresije

5.2.5 – JPEG kompresija

JPEG kompresija se sastoji iz tri faze:

1. **Diskretne kosinusne transformacije** (DCT) - predstavlja deljenje slike na blokove od po 8 x 8 piksela. DCT vrši pretvaranje ove matrice u istu takvu matricu ali sa nekim drugim vrednostima koje su sada prilagođene za kopresovanje. Tu se uzima u obzir: dodirne tačke i predviđa se šta može da bude kao i nijanse koje čovekovo oko može da registruje.
2. **Kvantizacije** - obezbeđuje način za ignorisanje malih promena u slici koje neće moći da se registruju pa nema potreba da se one i prenose.
3. **Faza kodiranja** - da se lineralizuju dvodimenzionalne matrice i izvrši njihovo kompresovanje radi prenosa. Koristi se Run-Length kodiranje.

5.2.6 – MPEG kompresija

- MPEG je standard za kompresiju pokretnih slika odnosno videa Upotrebljava slične tehnike kao JPEG.
- Koristi se činjenicom da su slike, koje slede jedna za drugom, a deo su nekog video snimka, u mnogo čemu slične.
- Nedostaci - potrebno je puno proračuna za generisanje komprimovane sekvence
- Postoji više tih standarda MPEG-1 do MPEG -7.

Postoje tri različita tipa kadra kod MPEG -a:

1. **I kadar** (unutrašnji kadar) – samosadržinski kadar koji predstavlja JPEG kodiranu sliku.
2. **P kadar** (predviđeni kadar) – ovaj kadar sadrži kodiranu informaciju o razlikama o tekućem i prethodnom kadru.
3. **B kadar** (bidirekcioni kadar) – sličan P kadru osim što je intrpoliran između prethodnog i budućeg kadra.

Tipična sekvenca kod MPEG-a se sastoji od sledećih kadrova:

I – B – B – P – B – B - I

5.2.6 – H.261 i H.263 standard

H.261 STANDARD

- Namenjen je standardizaciji prenosa slike kroz širokopojasnu digitalnu telefonsku mrežu (ISDN)
- Poznat je i kao standard **p×64 Kb/s** jer se za prenos koristi p ISDN kanala kapaciteta 64 Kb/s.
- Moguće vrednosti parametra p leže u granicama $1 \leq p \leq 30$.
- Uglavnom je namenjen za videofoniju i videokonferencije.

H.263 STANDARD

- Standard H.263 je namenjen standardizaciji prenosa slike po standardnim telefonskim komutiranim linijama pri bitskom protoku ispod 64 Kb/s, koji nije bio pokriven nijednim ranijim standardom
- Nastao je modifikacijama postojećeg H.261 standarda.
- Razlika između H.261 i H.263 standarda je u ciljnom bitskom protoku.
- H.261 se koristi za prenos slike iznad **64 Kb/s**, dok je H.263 se koristi ispod 64 Kb/s, najčešće pri protoku od **22 Kb/s**

5.2.7 – MP3 kompresija

- Osnovne karakteristike MP3 formata, i glavni razlog njegove velike popularnosti su velika kompresija uz očuvanje kvaliteta zvuka.
- MP3 odstranjuje uglavnom delove koji nisu primetni ljudskom uvu.
- Raspon ljudskog sluha se kreće između 20Hz–20KHz, a najosetljivije je između 2 - 4 KHz.
- MP3 fajlovi su CD kvaliteta, a zauzimaju 10 do 12 puta manje prostora nego ekvivalentni CD formati WAV/AIFF
- MP3 je skraćenica od ***MPEG-1 Audio Layer 3***.
- MPEG dopušta tri različita sloja audio kompresije.
- Slojevi se razlikuju po složenosti kodiranja, koeficijentima kompresije i rezultujućem kvalitetu zvuka, i to na sledeći način:

Sloj 1-koefic.kopresije 4:1, a zvuk se reprodukuje na brzini od 192 Kb/s.

Sloj 2-koefic.kopresije 8:1, a zvuk se reprodukuje na brzini od 128 Kb/s.

Sloj 3-koefic.kopresije 12:1, a zvuk se reprodukuje na brzini od 64 Kb/s.

5.3 – Integritet podataka

- Problem ko će da garantuje efikasne i sigurne komunikacije.
- Postavlja se pitanje kako utvrditi da je informacija koja se prenosi verodostojna i ista kao i ona koja je poslata.
- Posebno pitanje se postavlja da li je moguće da se deo informacije, koja je pogrešno primljena, ispravi bez ponovnog slanja.
- Pouzdanost podataka je vrlo bitna činjenica a nekada i neophodna.
- **Detekcija grešaka** - mogućnost detektovanja promena u toku prenosa
- Ponovno slanje poruka nekada nije izvodljivo (kod *real-time* sistema)
- Dve vrste grešaka mogu da nastupe: pogrešno primljen jedan bit, i pogrešno primljena grupa (paket) bitova.
- Paketske greške su učestalije i složenije za razrešavanje.
- Neophodno je da se koriste tehnike za detekciju greške.
- Svaka od ovih tehnika radi na sledećem principu: sekvenci bitova dodaju se bitovi koji čine kod za detekciju grešaka na predajnoj strani.
- Kod se izračunava kao funkcija drugih ostalih bitova koji se šalju.
- Uobičajeno za grupu od k bitova, algoritam za detekciju grešaka proizvodi kod za detekciju greške n−k bitova, gde je $(n-k) < k$.

5.3.1 – Jednostavne tehnike za detekciju greške

Provera parnosti

- Provera parnosti predstavlja najjednostavniju tehniku
- To znači da parna i neparna parnost podrazumeva dodavanje još jednog bita koji predstavlja "1" ili "0". Taj dodatni bit se naziva bit parnosti.
- Ukoliko se radi o parnoj parnosti vrednost ovog bita se selektuje tako da ukupan broj jedinica, uključujući i taj bit parnosti, bude paran, a ako se radi o neparnoj parnosti broj jedinica treba da bude neparan.
- Ova provera parnosti detektuje greške samo u jednom bitu što je u realnim uslovima veoma retko (50 % uspešnosti).
- **Navalna greška** (*burst error*) koja predstavlja grešku u više bitova.
- Osnova za složenije tehnike kod korekcije greške (Hamingovi kodovi) gde se vrši grupisanje bitova u grupe pa se na tom delu vrši provera
  1. horizontalna i vertikalna provera,
  2. ciklična provera pariteta gde se koriste dva bita parnosti – jedan za neparne bitove a drugi za parne bitove u poruci,
  3. longitudalna provera parnosti kod koje se vrši provera parnosti svih bitova na istoj poziciji karaktera u celoj poruci.

5.3.1 – Jednostavne tehnike za detekciju greške

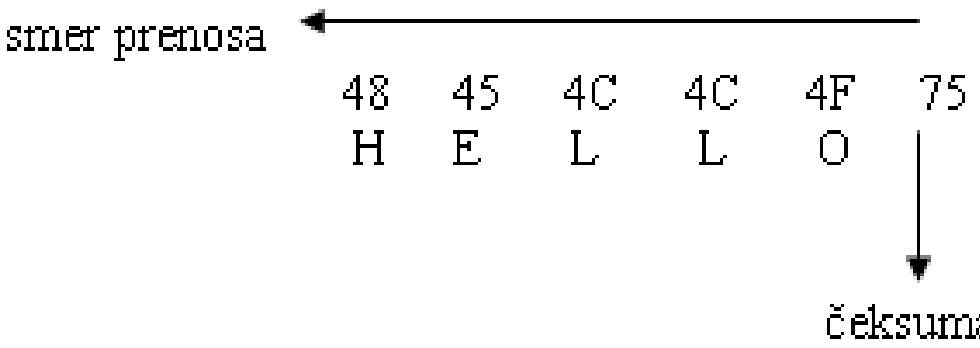
Primer: horizontalna i vertikalna provera

	Bitovi parnosti	ASCII kod	Karakter
	0	1000010	B
	1	1000101	E
	1	1001111	O
	0	1000111	G
	1	1010010	R
	0	1000001	A
	0	1000100	D
reč parnosti	1	1011000	

5.3.1 – Jednostavne tehnike za detekciju greške

Ček suma

- Podrazumeva grupisanje svih bitova u grupe od po 8, 16 ili 32 bita
- Svaka grupa se sada tretira kao celobrojna vrednost.
- Te se vrednosti sada sabiraju (suma po modulu 2), tako da daju novi 8, 16 ili 32-bitni podatak – **čeksumu** (ona se formira tako da se bit najveće težine koji izlazi iz 8,16 ili 32 bita ignoriše).
- Dobijena vrednost se pridružuje podacima koji se šalju.
- Na prijemnoj strani radi se isti takav postupak i ako se ček sume ne poklapaju to znači da je došlo do greške u prijemu podataka.
- Ovo je nešto bolji način od kontrole pariteta jer može da detektuje navalne greške ali ni ovo nije pouzdano jer može da se desi da je čeksuma ista a da su podaci loše primljeni (+1 i -1 u različitim karakterima)



5.3.1 – CRC (Cyclic Redundancy Check)

- Polazi od činjenice da svaka povorka bitova može da se tretira kao polinom sa koeficijentima 0 i 1.
- Ram od *k* bitova se smatra listom koeficijenata polinoma sa *k* članova počevši od x^{k-1} do x^0. Za takav polinom se kaže da je stepena *k-1*.

Primer: *110001 ima 6 bitova i predstavlja šestočlani polinom sa koeficijentima 1, 1, 0, 0, 0 i 1 :* $x^5 + x^4 + x^0$.

Počiva na:

1. aritmetici po modulu 2 (ne postoji prenos kod sabiranja)
2. deljenju polinoma
3. niz bitova bloka koji se prenosi se posmatra kao niz koeficijenata polinoma, npr. $a_n a_{n-1} \ldots a_1 a_0$ odgovara polinomu:

$$M(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

5.3.1 – CRC (Cyclic Redundancy Check)

Postupak kodiranja:

1. Na niz koji se šalje (polinom ***M(x)***) dodaje se onoliki broj nula koliki je stepen generatora polinoma.
2. Taj niz se deli sa unapred definisanim polinomom ***G(x)*** - generator polinoma. Ostatak tog deljenja daje nam polinom ***R(x)***
3. Definišimo sada polinom ***T(x)*** tako da je ***T(x)=M(x)-R(x)*****.** Ovo oduzimanje ne predstavlja ništa drugo nego zamenu prethodno dodatih nula polinomu ***M(x)*** sa nizom bitova koji odgovaraju ***R(x)***.
4. Sada se niz bitova, polinom ***T(x)*****,** stvarno prenosi ka prijemniku.

Postupak dekodiranja:

1. primljena polinomijalna kodna reč se deli sa *G(x)*-generator polinoma
2. ako je ostatak deljenja 0, nema grešaka pri prenosu
3. ako ostatak nije nula, postoje greške pri prenosu

- Ideja je da se doda kontrolna suma (c*hecksum*) na kraj rama na takav način da je polinom koji je predstavljen ramom sa kontrolnom sumom deljiv sa *G(x)*.

5.3.1 – CRC (Cyclic Redundancy Check)

- Postoje greške koje se ovako ne mogu otkriti, ali se dobrim izborom polinoma generatora njhov broj smanjuje
- Dobri polinomi generatori su:

  CRC-12= $x^{12}+x^{11}+x^3+x^2+x+1$

  CRC-16 = $x^{16}+x^{15}+x^2+1$

  CRC-CCITT = $x^{16}+x^{12}+x^5+1$

  CRC-CCiTT = $x^{32}+x^{26}+x^{23}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+1$

CRC može da detektuje:

- sve greške na neparnom broju bitova
- sve 2-bitne greške
- navalne greške čija je dužina r manja od stepena polinoma G(x)
- navalne greške čija je dužina jednaka r+1 sa verovatnoćom od $(2^{r-1}-1)/2^{r-1}$
- navalne greške čija je dužina veća od r+1 sa verovatnoćom od $(2^r-1)/2^r$
- To znači da CRC-32 polinom detektuje greške čija je dužina veća od 33 sa verovatnoćom od $(2^{32}-1)/2^{32}$ što isnosi 99,99999 tačnosti

5.3.1 – CRC (Cyclic Redundancy Check)

Primer:

Odrediti niz bitova koji šalje ako se šalju bitovi 11100110 a generator polinoma je $G(x)=x^4+x^3+1$.

Rešenje:

dodajemo 4 bita 0 (množimo polinom sa x^4) i delimo:

11100110**0000**
-11001
1011100000
-11001
111000000
-11001
10100000
-11001
1101000
-11001
0110

Na originalne bitove dopisujemo ostatak 0110 i dobijamo: 11100110**0110**

Provera po prijemu:

delimo:

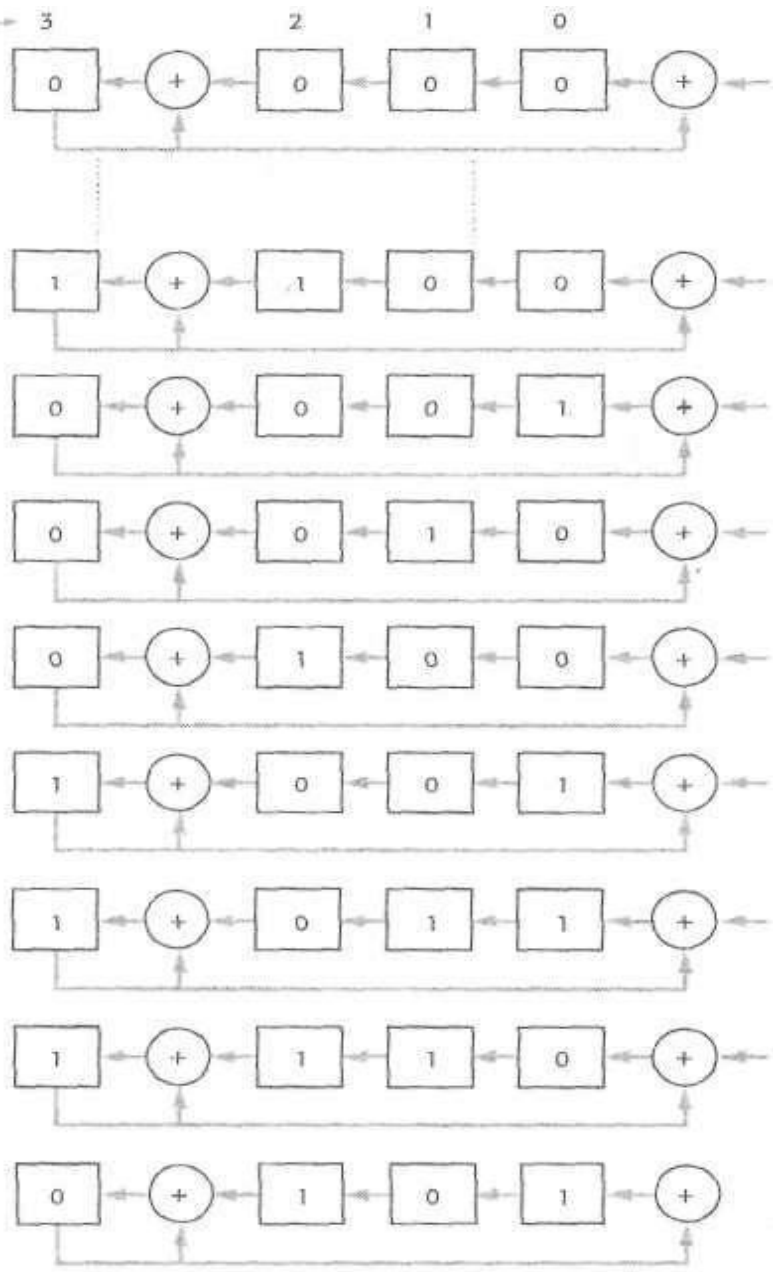
11100110**0110**
-11001
1011100110
-11001
111000110
-11001
10101110
-11001
1100110
-11001

ostatak je 0, dakle prenos je ispravan: 1100110

5.3.1 – CRC (Cyclic Redundancy Check)

1. Broj bitskih pozicija u registru je r, gde je krajnja desna pozicija član b_0 a krajnja leva član b_{r-1}.
2. Isključivo ILI kolo se nalazi desno od bilo koje pozicije kojoj je pridružena vrednost $b_i = 1$
3. Niz bitova se uvodi u registar jedan po jedan počevši od desne strane.
4. Kada se uvede novi bit, svi bitovi koji se nalaze u registru pomeraju se za 1 mesto u levo poštujući pravila sabiranja ***isključivo ili*** gde se to kolo nalazi.
5. Bit sa krajnje leve pozicije se propušta kroz sva ***isključivo ili*** kola formirajući na taj način drugi operand za ta kola.

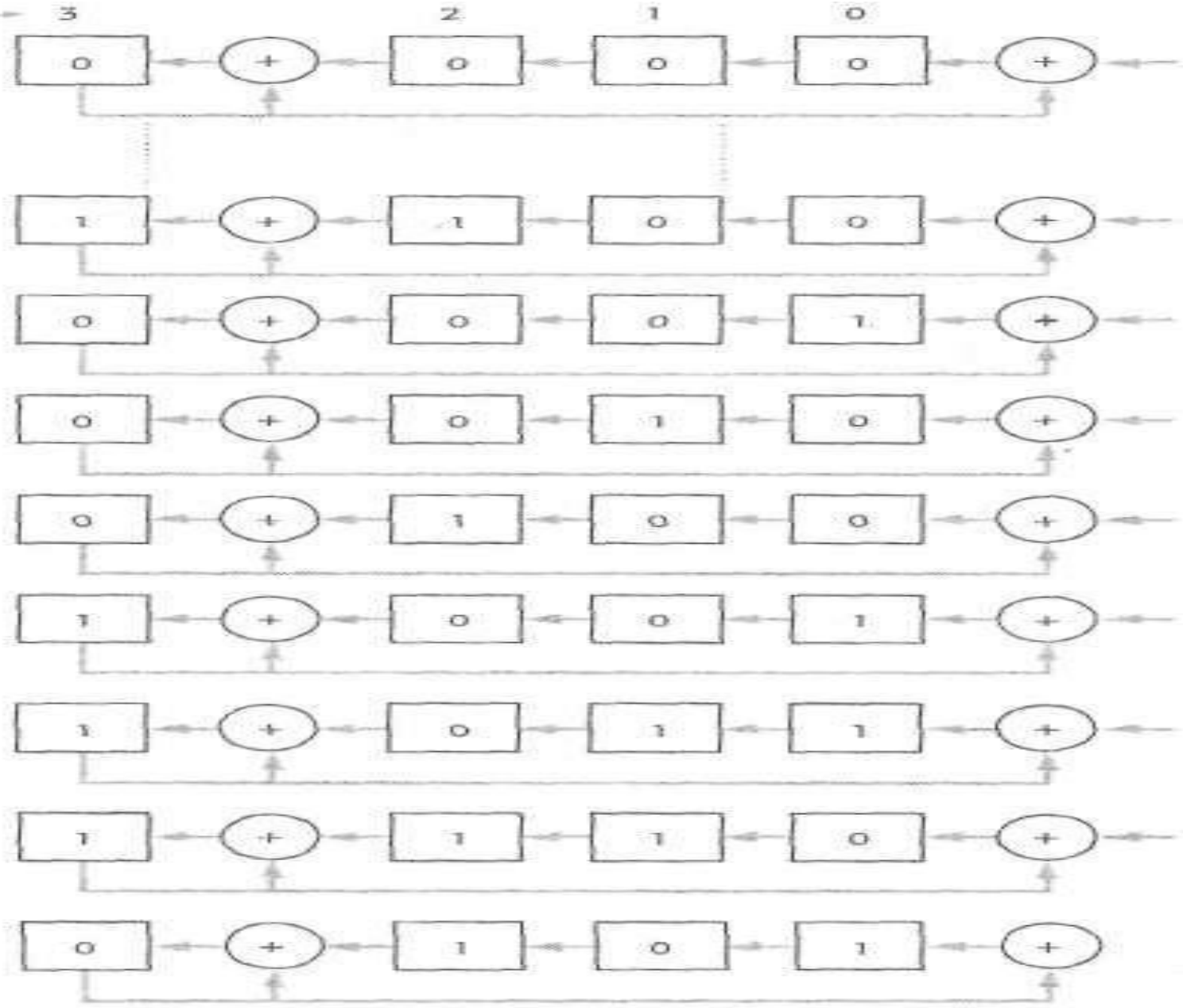
Na slici prikazana je deoba dva polinoma M(x): $x^{10}+x^9+x^3+x$ i $G(x)=x^4+x^3+1$,



5.3.1 – CRC (Cyclic Redundancy Check)

Polinom M(x)
$x^{10}+x^9+x^3+x$
11000001010

Generator
Polinoma G(x)
x^4+x^3+1
11001



5.4 – Korekcija grešaka

- Kada se greške detektuju postoje dva rešenja: ponovno slanje te iste poruke ili pokušaj restauriranja tj. ispravljanja oštećenog okvira.
- Najprostiji kod za otkrivanje i korekciju grešaka je *Hamingov kod*
- Hamingov kod podrazumeva umetanje višestrukih bitova parnosti u niz bitova pre nego što se on pošalje.

n (broj provera parnosti)	Broj poslatih bitova	2^n broj kombinacija
1	9	2
2	10	4
3	11	8
4	12	16

- Svaki bit parnosti uspostavlja parnu parnost za selektovane pozicije.

Podaci koji se šalju: $m_1\ m_2\ m_3\ m_4\ m_5\ m_6\ m_7\ m_8$

Hamingov kod: $p_1\ p_2\ m_1\ p_3\ m_2\ m_3\ m_4\ p_4\ m_5\ m_6\ m_7\ m_8$

p_1-parna parnost na pozicijama 1,3,5,7,9,11

p_2-parna parnost na pozicijama 2,3,6,7,10,11

p_3-parna parnost na pozicijama 4,5,6,7,12

p_4-parna parnost na pozicijama 8,9,10,11,12

5.4 – Hamingov kod

Primer: *Treba da prenesemo podatke* **0 1 1 0 0 1 1 1**

Hamingov kod je tada: **0 1 0 1 1 1 0 1 0 1 1 1**

Pretpostavimo da je primljeni niz bitova: 0 1 0 1 0 1 0 1 0 1 1 1

Da bi znali da li je poruka dobro primljena potrebno je da proverimo bitove parnosti u poruci koju smo dobili a koji se nalaze na pozicijama 2^n. Nakon provere dobijamo sledeće rezultate:

Bitovi na pozicijama 1, 3, 5, 7, 9, 11 (p_1), provera parnosti je neuspešna

$\mathbf{p_1 = 1}$

Bitovi na pozicijama 2, 3, 6, 7, 10, 11 (p_2), provera parnosti je uspešna

$\mathbf{p_2 = 0}$

Bitovi na pozicijama 4, 5, 6, 7, 9, 12 (p_3), provera parnosti je neuspešna

$\mathbf{p_3 = 1}$

Bitovi na pozicijama 8, 9, 10, 11, 12 (p_4), provera parnosti je uspešna

$\mathbf{p_4 = 0}$

Na osnovu toga zaključujemo da je greška je nastala na poziciji 0101 tj. na 5 bitu gde je primljena logička 0 a trebalo je da bude 1 što na osnovu ovoga možemo da ispravimo.

5.4 – Korigovanje višestrukih grešaka

- Za korigivanje višestrukih grešaka postoje rešenja koja su dosta komplikovana i dovoljno je da znamo da su to ***BHC***(*Bose-Chaundhuri-Hocquenghem*) kodovi i ***Reed-Solomonovi*** kodovi koji predstavljaju podklasu *BHC* kodova.
- Obe metode se koriste konceptom kodnih reči, tj. kolekcije od *n* bitova iza koji stoji *m* kontrolnih bitova greške koji se izračunavaju na osnovu bitova podataka.
- Ključni koncept ovih kodova podrazumeva da postoji određeno **rastojanje** između dve kodne reči, tj. da postoje različiti bitovi u tim ključnim rečima. Na primer, dva Hamingova koda 0101-1101-0111 i 1001-1001-0111 imaju rastojanje 3, jer se razlikuju u tri bita (prvi, drugi i šesti bit). Svaki skup kodnih reči ima **minimalno rastojanje**.
- U opštem slučaju, ako je *d* minimalno rastojanje, ovaj metod može da detektuje sve greške koje utiču na manje od *d* bitova (takva promena daje neispravnu kodnu reč) i može da ispravi sve greške koje utiču na manje od *d/2* bitova.

Hvala na pažnji !!!

Pitanja

? ? ?