Otkrivanje i korekcija grešaka

Pri zapisu ili prenosu podataka ponekad se može dogoditi da se neki od bitova greškom promene, tj. invertuju. Ovo je obično rezultat nekog šuma ili smetnje. Primeri takvih situacija su prenos podataka između dva računara preko računarske mreže, kao i prenos podataka unutar istog računarskog sistema, preko magistrale. Takođe, do greške može doći i prilikom čuvanja podataka u memoriji (tipičan primer su dinamičke RAM memorije, gde se može dogoditi da usled raznih uticaja (električnih, radioaktivnih, kosmičkih) dođe do promene vrednosti nekog sačuvanog bita). U takvim situacijama oštećeni podatak prestaje da bude upotrebljiv, jer jedan promenjeni bit može da predstavlja potpuno drugi broj, karakter, instrukciju, i sl. Zbog toga je potrebno omogućiti otkrivanje ovakvih grešaka, a po mogućstvu i njihovo otklanjanje, tj. korekciju na licu mesta.

Algoritmi za otkrivanje grešaka nam omogućavaju samo da konstatujemo da je do greške došlo, bez mogućnosti da saznamo koji su bitovi invertovani, kako bismo na licu mesta popravili grešku. Zbog toga je u takvim situacijama neophodno ceo podatak preneti ponovo (u slučaju da je greška nastala prilikom prenosa podatka), ili konstatovati da oporavak od greške nije moguć (npr. kod greške u memoriji). Ovakvi algoritmi se obično koriste tamo gde ponovni prenos podataka nije skup, kao i u slučajevima gde se greške relativno retko javljaju. *Algoritmi za korekciju grešaka*, sa druge strane, omogućavaju da se na licu mesta rekonstruiše ispravan podatak. Ovi algoritmi zahtevaju veći broj dodatnih bitova koji se pridružuju poruci i koriste se tamo gde se greške javljaju relativno često (pa bi učestalo ponovno slanje drastično uticalo na performanse sistema), kao i u slučajevima gde ponovno slanje podatka nije moguće (npr. u slučaju memorijskih grešaka, nemamo odakle da iznova dobijemo isti podatak).

U nastavku ovog teksta dajemo kratak teorijski uvod u otkrivanje i korekciju grešaka, a zatim prezentujemo i neke često korišćene algoritme za otkrivanje i korekciju grešaka.

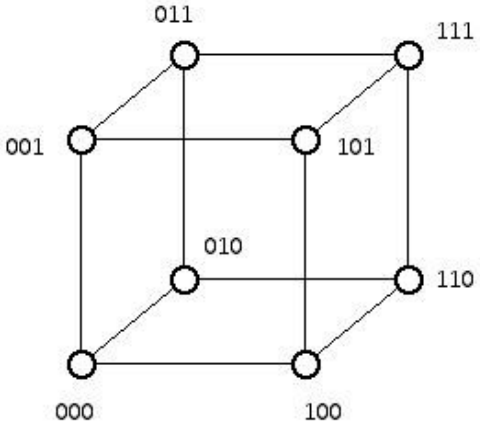
Teorijski uvod

Neka je data *poruka* P koja se sastoji iz m bitova (to može biti podatak koji se šalje preko nekog komunikacionog kanala, ili podatak koji se čuva u memoriji). Osnovna ideja svih algoritama za otkrivanje i korekciju grešaka je da se definiše neka funkcija kodiranja $\mathcal{K}$ koja svakoj poruci P pridružuje niz bitova $C = \mathcal{K}(P)$ neke fiksne dužine $n = m + r$. Nisku C nazivamo *kodna reč* pridružena poruci P. Kodna reč se obično dobija tako što se na m bitova poruke P doda još r tzv. *kontrolnih bitova*. Ovi bitovi se obično dopisuju na kraj poruke, ali se kod nekih algoritama mogu umetati i između bitova poruke P. Vrednosti kontrolnih bitova su jednoznačno određene za svaku poruku P. Skup svih ovako dobijenih kodnih reči je *kod* $\mathcal{K}$. Primetimo da je

ukupan broj niski od n bitova jednak 2^n, ali od svih takvih niski samo njih 2^m predstavljaju ispravne kodne reči koda $\mathcal{K}$, jer svakoj poruci P odgovara tačno jedna kodna reč $C = \mathcal{K}(P)$ (a poruka dužine m ima 2^m). Ukoliko se prilikom prenosa dobije neka od $2^n - 2^m$ *neispravnih kodnih reči*, tada je jasno da je došlo do greške.

Pod *Hamingovim rastojanjem* dve niske bitova C_1 i C_2 iste dužine n (u oznaci $d(C_1, C_2)$) podrazumevamo broj bitskih pozicija na kojima se ove dve niske razlikuju. Na primer, ako je $C_1 = 10100011$, a $C_2 = 10101001$, tada je $d(C_1, C_2) = 2$, jer se ove dve niske razlikuju na pozicijama 5 i 7 (pozicije brojimo sa leva u desno počev od 1).[1]

Hamingovo rastojanje između binarnih niski se najbolje može ilustrovati geometrijski, pomoću hiperkocke. *Jedinična hiperkocka* u n-dimenzionom prostoru ima 2^n temena kojima pridružujemo binarne niske dužine n. Ove niske odgovaraju koordinatama temena u koordinatnom sistemu takvom da jedno od temena hiperkocke predstavlja koordinanti početak (označimo ga sa O), a ose koordinatnog sistema su određene ivicama koje su susedne temenu O (njih ima n). Na primer, za niske dužine 3 možemo posmatrati 3-dimenzionu kocku na donjoj slici (koordinatni početak je teme 000, a ose su ivice kocke susedne tom temenu).



Hamingovo rastojanje između dve niske se sada može razumeti kao dužina najkraćeg puta po ivicama kocke između odgovarajućih temena kocke. Na primer, rastojanje između niske 101 i 110 je 2, jer je najkraće rastojanje

[1] Izračunavanje Hamingovog rastojanja u računaru se može jednostavno realizovati tako što se najpre izvrši *bitovska ekskluzivna disjunkcija* dve niske, a zatim se prebroje jedinice u dobijenom rezultatu.

između odgovarajuća dva temena jednako 2 (od 101 do 110 možemo stići u dva koraka, ili preko temena 111, ili preko temena 100).

Pod *k-okolinom* ($k > 0$) binarne niske C podrazumevamo skup $K(C,k) = \{C' \mid d(C,C') \leq k\}$. Drugim rečima, to je skup svih niski iste dužine koje su na rastojanju najviše k od niske C. Primetimo da sama niska C pripada svakoj svojoj k-okolini. Na primer, 1-okolina niske 101 je skup $\{101, 001, 111, 100\}$. Dakle, 1-okolina neke niske, pored same te niske, uključuje i niske koje odgovaraju susednim temenima hiperkocke (videti gornju sliku).

Neka je C originalna (*izvorna*) kodna reč (poslata preko komunikacionog kanala, ili sačuvana u memoriju), a C' kodna reč koja je *očitana* (na izlazu komunikacionog kanala, ili iz memorije). Ukoliko ove dve reči nisu identične (tj. ako im Hamingovo rastojanje nije 0), tada imamo grešku. Hamingovo rastojanje $d = d(C,C')$ nazivamo *obimom greške*. Pojedini algoritmi za otkrivanje i korekciju grešaka su konstruisani tako da mogu otkriti (odnosno korigovati) samo greške do određenog obima. Naime, pod *Hamingovim rastojanjem koda* $\mathcal{K}$ (u oznaci $d(K)$) podrazumevamo najmanje Hamingovo rastojanje između neke dve različite ispravne kodne reči koda $\mathcal{K}$. Na primer, ako imamo trobitni kod K koji sadrži kodne reči 000, 101, 110 i 011, tada je $d(K) = 2$, jer nikoje dve kodne reči nisu na rastojanju manjem od 2, a postoje dve kodne reči (npr. 101 i 110) koje su na rastojanju 2. Da bi kod $\mathcal{K}$ omogućio otkrivanje grešaka obima najviše d, potrebno je da nikoje dve ispravne kodne reči koda $\mathcal{K}$ nemaju Hamingovo rastojanje manje ili jednako od d (tj. Hamingovo rastojanje koda $\mathcal{K}$ treba da bude najmanje $d + 1$). Ovim je garantovano da ukoliko se dogodi greška obima najviše d, dobijena niska neće biti ispravna kodna reč, jer u d-okolini svake ispravne kodne reči nema drugih ispravnih kodnih reči. Slično, da bi kod $\mathcal{K}$ omogućavao korekciju grešaka obima najviše d, potrebno je da nikoje dve ispravne kodne reči koda $\mathcal{K}$ nemaju Hamingovo rastojanje manje ili jednako od $2d$ (tj. Hamingovo rastojanje koda $\mathcal{K}$ mora da bude najmanje $2d + 1$). Ovim se postiže da d-okoline svake dve ispravne kodne reči budu međusobno disjunktne, pa ukoliko se dogodi greška obima najviše d, dobijena niska će biti u d-okolini polazne kodne reči, a van d-okolina drugih ispravnih kodnih reči (tj. biće najbliža polaznoj ispravnoj kodnoj reči). Ovo svojstvo nam omogućava da rekonstruišemo originalnu kodnu reč.

U naredna dva odeljka ćemo predstaviti primere algoritama za otkrivanje i korekciju grešaka obima 1. U poslednjem odeljku ćemo prikazati jedan algoritam za otkrivanje grešaka većeg obima.

Algoritam za otkrivanje grešaka obima 1

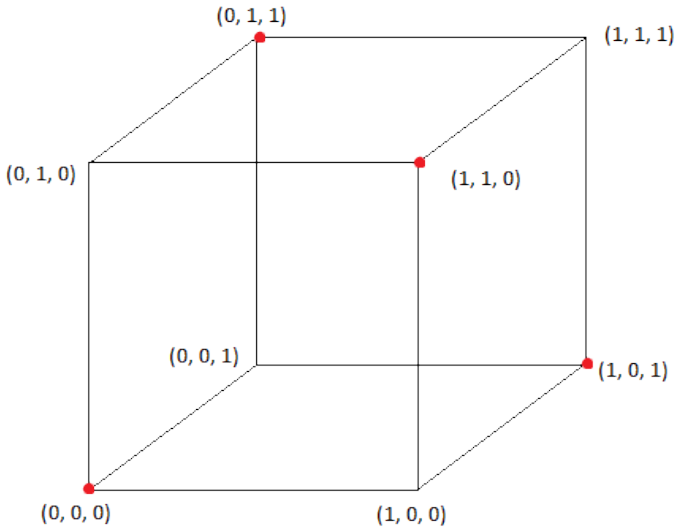
Da bismo bili u mogućnosti da otkrijemo bilo koju grešku obima $d = 1$, potrebno je da konstruišemo kod $\mathcal{K}$ čije je Hamingovo rastojanje bar $d + 1 = 2$. Da bismo ovo postigli, dovoljno je da dodamo jedan kontrolni bit

koji omogućava da se između svake dve ispravne kodne reči ubaci po jedna neispravna. Najjednostavniji način da se ovo postigne je da se zahteva da u dobijenoj kodnoj reči $C = \mathcal{K}(P)$ (koja se dobija tako što se dodatni kontrolni bit dopiše na m bitova poruke P) ukupan broj bitova čija je vrednost 1 bude paran. Ovakav postupak otkrivanja grešaka naziva se *kontrola parnosti*. Na primer, ako imamo poruku $P = 10100001$ ($m = 8$), možemo primetiti da je broj jedinica u zapisu poruke neparan. Zbog toga ćemo dodati još jednu jedinicu na kraj poruke i dobićemo kodnu reč $C = 101000011$ koja sada ima $8 + 1 = 9$ bitova, među kojima je paran broj jedinica. Sa druge strane, da smo imali poruku $P = 11001100$, tada bismo na kraj poruke dopisali 0, jer poruka P već ima paran broj jedinica u svom zapisu, pa dobijamo kodnu reč $C = 110011000$ koja takođe ima paran broj jedinica. Dakle, ispravne kodne reči su one kodne reči koje imaju paran broj jedinica, dok su neispravne kodne reči one koje imaju neparan broj jedinica u svom zapisu.

Pretpostavimo sada da je originalna kodna reč $C = 101000011$, a da je očitana kodna reć $C' = 111000011$ (primetimo da je $d(C, C') = 1$). Očitana kodna reč C' je neispravna, jer ima neparan broj jedinica u svom zapisu, pa je zato odbacujemo kao pogrešnu (tj. naš algoritam prepoznaje da je došlo do greške). Sa druge strane, ako je očitana kodna reč $C'' = 110000011$ (primetimo da je $d(C, C'') = 2$), tada će naš algoritam (pogrešno) protumačiti dobijenu kodnu reč kao ispravnu, jer njen zapis sadrži paran broj jedinica. Dakle, vidimo da algoritam zasnovan na kontroli parnosti u opštem slučaju ne može da otkrije greške obima većeg od 1.[2]

Na kraju ovog odeljka, poslužimo se opet ilustracijom pomoću hiperkocke na donjoj slici (slučaj $m = 2$, tj. kodne reči su dužine 3):

[2]Primetimo da ukoliko bi obim greške bio neparan (npr. 3 ili 5), tada bi dobijena kodna reč imala neparan broj jedinica u svom zapisu, pa bi greška bila otkrivena. Dakle, algoritam zasnovan na kontroli parnosti može da otkrije i neke greške obima većeg od 1 (preciznije, sve greške neparnog obima), ali ne sve takve greške. Uopšte, većina algoritma za otkrivanje grešaka garantuju otkrivanje nekih tipova grešaka, ali često pored toga mogu otkriti i neke greške koje nisu iz tog skupa grešaka.



Na gornjoj slici, crvena temena odgovaraju ispravnim kodnim rečima, dok ostala temena odgovaraju neispravnim kodnim rečima. Sa slike se vidi da za svaku ispravnu kodnu reč u njenoj 1-okolini nema drugih ispravnih kodnih reči. Ovo svojstvo omogućava otkrivanje svih grešaka obima 1. Sa druge strane, 1-okoline ispravnih kodnih reči nisu disjunktne. Zbog toga nemamo mogučnost korekcije grešaka obima 1. Na primer, ako je očitana kodna reč 100, mi ćemo moći da utvrdimo da je neispravna (jer ima neparan broj jedinica), ali nećemo moći da utvrdimo koja je originalna ispravna kodna reč, jer se kodna reč 100 istovremeno nalazi u 1-okolinama čak 3 ispravne kodne reči (101, 110, 000). Očigledno je da će nam za korekciju grešaka obima 1 biti potrebno više kontrolnih bitova, kako bismo povećali Hamingovo rastojanje koda. Jedan takav postupak prikazujemo u sledećem odeljku.

Algoritam za korekciju grešaka obima 1

Kao što je ranije rečeno, da bismo mogli da korigujemo svaku grešku obima $d = 1$, potrebno je da rastojanje koda bude najmanje $2d + 1 = 3$, kako bismo mogli da postignemo da 1-okoline svake dve ispravne kodne reči budu međusobno disjunktne. Ispostavlja se da broj dodatnih kontrolnih bitova koji nam ovako nešto omogućava nije unapred fiksiran, već zavisi od dužine poruke m (primetimo da je kod otkrivanja grešaka obima 1 uvek bio dovoljan jedan kontrolni bit, bez obzira na dužinu poruke). Naime, u 1-okolini svake kodne reči C dužine $n = m + r$ ima $n + 1$ kodnih reči (kodna reč C, kao i sve kodne reči koje nastaju od C tako što se promeni tačno jedan bit). Na primer, za kodnu reč $C = 101$, njenu 1-okolinu čine kodne reči $101, 001, 111, 100$.

Da bismo omogućili da 1-okoline ispravnih kodnih reči budu disjunktne (a imamo ukupno 2^m ispravnih kodnih reči), sledi da ćemo morati da imamo bar $(n+1)\cdot 2^m$ različitih kodnih reči (ispravnih i neispravnih). Kako binarnih niski dužine n ima tačno 2^n, sledi da mora važiti relacija $(n+1)\cdot 2^m \leq 2^n$, odnosno $m+r+1 \leq 2^r$, imajući u vidu da je $n = m+r$. Na primer, za $m = 1$ potrebno je da važi relacija $r + 2 \leq 2^r$. Najmanje r koje ovo zadovoljava je $r = 2$. Dakle, da bismo bili u mogućnosti da popravimo svaku grešku obima 1 u poruci dužine 1 (tj. jednobitnoj poruci), potrebno je da dodamo još dva kontrolna bita. Na sreću, ispostavlja se da broj kontrolnih bitova ne raste tako drastično sa povećavanjem dužine poruke (asimptotski, r raste približno logaritamskom brzinom u odnosu na m). Ovo se može ilustrovati sledećom tabelom:

m	1	2	3	4	5	6	...	11	12	13	...	26	27	...	57	58	...
r	2	3	3	3	4	4	...	4	5	5	...	5	6	...	6	7	...

Postupak za korekciju grešaka obima 1 koji se standardno koristi u praksi je poznat pod nazivom *Hamingov kod*. Koristićemo oznaku (n, m) za označavanje Hamingovog koda koji se koristi za kodiranje poruke dužine m, pri čemu je ukupan broj bitova kodne reči zajedno sa kontrolnim bitovima jednak n (tj. $n = m + r$). Kod Hamingovog koda, kontrolni bitovi se ne dopisuju na kraj poruke, već se umeću između bitova poruke. Pritom, kontrolni bitovi se uvek nalaze na pozicijama čiji su redni brojevi stepeni dvojke, tj. na pozicijama 1, 2, 4, 8, 16, ... (pozicije se broje sa leva u desno, počev od 1). Na primer, u Hamingovom kodu $(7, 4)$, imamo 7 bitova u kodnoj reči, pri čemu su bitovi na pozicijama 1, 2 i 4 kontrolni bitovi, dok su ostala 4 bita (na pozicijama 3, 5, 6 i 7) redom bitovi poruke. Primetimo da se na ovaj način postiže da zaista u kodnoj reči od n bitova imamo odgovarajući broj kontrolnih bitova r određen relacijom $m + r + 1 \leq 2^r$. Zaista, može se uočiti da za svako m i odgovarajuće minimalno r koje zadovoljava gornju relaciju važi da je broj stepena dvojke koji su manji ili jednaki od $n = m+r$ upravo jednak r (to su stepeni $2^0, 2^1, \ldots, 2^{r-1}$). Otuda, ako u kodnoj reči za kontrolne bitove rezervišemo one pozicije čiji su redni brojevi jednaki stepenima dvojke, uvek ćemo imati tačno onoliki broj kontrolnih bitova koliko je potrebno da obezbedimo mogućnost korekcije grešaka obima 1.

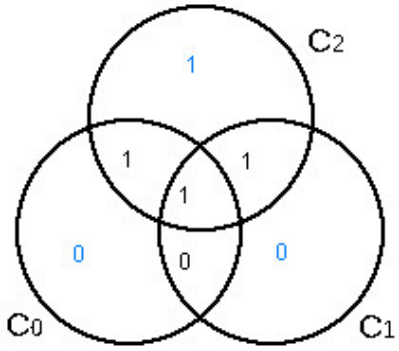
Prilikom određivanja Hamingovog koda za datu poruku P dužine m, najpre ćemo bitove poruke upisati u bitove kodne reči, preskačući pritom pozicije čiji su redni brojevi stepeni dvojke (jer su te pozicije rezervisane za kontrolne bitove). Na primer, za poruku $P = 0111$ dužine $m = 4$ imaćemo kodnu reč $C = \square\square 0 \square 111$, gde simbol $\square$ označava da na toj poziciji treba upisati odgovarajući kontrolni bit. Kontrolni bit na poziciji 2^i određujemo iz uslova da broj bitova kodne reči sa vrednošću 1 koji se nalaze na pozicijama čiji redni brojevi u svom binarnom zapisu imaju jedinicu na poziciji

sa težinom 2^i mora biti paran. Na primer, kontrolni bit na poziciji $1 = 2^0$ kontroliše sve pozicije u kodnoj reči čiji redni brojevi u binarnom zapisu imaju jedinicu na najnižoj poziciji (to su pozicije 1, 3, 5, 7). Kontrolni bit na poziciji $2 = 2^1$ kontroliše pozicije čiji redni brojevi u binarnom zapisu imaju jedinicu na poziciji sa težinom 2 (to su pozicije 2, 3, 6, 7). Najzad, kontrolni bit na poziciji $4 = 2^2$ kontroliše sve pozicije čiji redni brojevi u svom binarnom zapisu imaju jedinicu na poziciji sa težinom 4 (to su pozicije 4, 5, 6, 7). Primetimo da svaki kontrolni bit uvek kontroliše i samog sebe. Takođe, jedan isti bit može biti kontrolisan od strane više kontrolnih bitova. Postupak kodiranja za naš primer ilustrovan je sledećom tabelom:

	C	**Pozicija**	C_2	C_1	C_0
C_0:	**0**	1	0	0	1
C_1:	**0**	2	0	1	0
	0	3	0	1	1
C_2:	**1**	4	1	0	0
	1	5	1	0	1
	1	6	1	1	0
	1	7	1	1	1

Dakle, kontrolni bit C_i se određuje tako da se obezbedi parnost broja jedinica u koloni **C** koje odgovaraju pozicijama kod kojih u koloni C_i imamo jedinicu. Na primer, u koloni C_1 imamo jedinice koje odgovaraju pozicijama 2, 3, 6 i 7 (to su upravo pozicije čiji redni brojevi u binarnom zapisu imaju jedinicu na poziciji sa težinom 2^1). Zato za određivanje kontrolnog bita C_1 moramo da prebrojimo koliko ima jedinica na tim pozicijama u kodnoj reči. Kako su bitovi na pozicijama 3, 6 i 7 redom 0, 1 i 1, to je broj jedinica već paran, pa kontrolni bit na poziciji 2 (tj. kontrolni bit C_1) treba da bude jednak 0. Slično se određuju i druga dva kontrolna bita. Kodna reč koja se dobija je $C = \mathcal{K}(P) = \mathbf{00}\mathbf{0}111$ (podebljanim fontom su označeni kontrolni bitovi).

Hamingov kod $(7, 4)$ se često ilustruje i pomoću *Venovih dijagrama* (videti sliku ispod koja odgovara prethodnom primeru). Svaki od skupova odgovara jednom kontrolnom bitu C_0, C_1 i C_2. U svaki od preseka upisujemo onaj bit poruke P koji je kontrolisan upravo onim kontrolnim bitovima čijih je skupova to presek. Same kontrolne bitove upisujemo u delove skupova koji ne pripadaju ni jednom od preseka. Svaki kontrolni bit se određuje tako da u odgovarajućem skupu imamo paran broj jedinica. Na ovaj način, svaki kontrolni bit kontrolisaće upravo one bitove koji se nalaze u njegovom skupu (samog sebe, i još tri bita poruke).

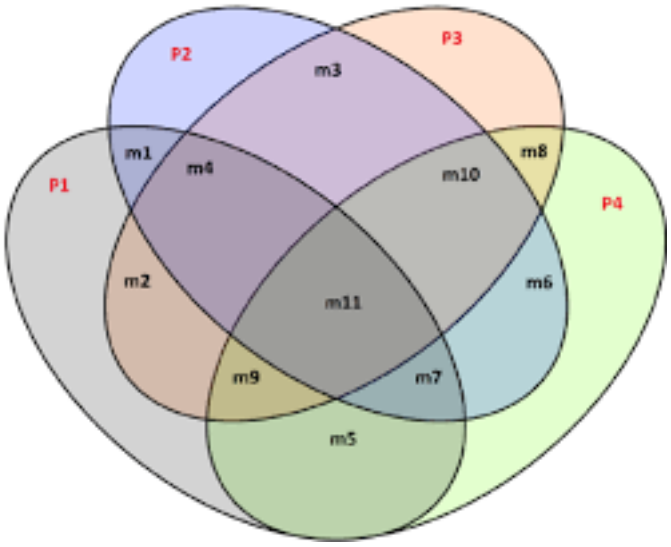


Ukoliko dođe do greške obima 1, tada Hamingov kod omogućava određivanje rednog broja pozicije na kojoj se u očitanoj kodnoj reči C' nalazi pogrešan bit. Ovo se postiže ponavljanjem istog postupka kao prilikom kodiranja (tj. prebrojavaju se jedinice na odgovarajućim pozicijama u kodnoj reči), pri čemu se utvrđuje za koje kontrolne bitove imamo neparan broj jedinica na pozicijama koje ti bitovi kontrolišu. Neka je C'_i bit čija je vrednost 1 ukoliko u kodnoj reči C' na pozicijama koje kontroliše bit C_i imamo neparan broj jedinica, a 0 u suprotnom. Tada će binarni zapis $C'_2C'_1C'_0$ određivati redni broj pozicije na kojoj se pojavila greška. Na primer, neka je u kodnoj reči $C = 0001111$ iz našeg primera bit na poziciji 6 invertovan, tj. neka je očitana kodna reč $C' = 0001101$. Formirajmo sličnu tabelu kao i malopre:

	C	**Pozicija**	C_2	C_1	C_0
$C'_0 = 0$:	0	1	0	0	1
$C'_1 = 1$:	0	2	0	1	0
	0	3	0	1	1
$C'_2 = 1$:	1	4	1	0	0
	1	5	1	0	1
Greška:	**0**	6	1	1	0
	1	7	1	1	1

Podebljanim fontom označen je pogrešan bit, a vrednosti C'_i su zapisane sa leve strane tabele. Ove vrednosti se izračunavaju prilikom očitavanja kodne reči, a na osnovu njih zaključujemo da je pozicija greške $C'_2C'_1C'_0 = 110_{(2)} = 6$. U slučaju ispravno očitane kodne reči svi C'_i bitovi bi imali vrednost nula.

Ilustracija zasnovana na Venovim dijagrama postaje manje upotrebljiva u slučaju dužih poruka, jer je tada broj kontrolnih bitova veći, pa je samim tim potrebno pregledno nacrtati više od 3 skupa, što nije tako jednostavno. Na primer, za $r = 4$ bismo imali sledeći Venov dijagram:



Sa druge strane, postupak dat gornjom tabelom lako je uopštiti i za poruke proizvoljne dužine. Na primer, ako imamo poruku $P = 1011010110101101$ dužine $m = 16$, potrebno je $r = 5$ kontrolnih bitova (dakle, imamo Hamingov kod $(21, 16)$. Određivanje Hamingovog koda prikazano je sledećom tabelom:

	C	**Pozicija**	C_4	C_3	C_2	C_1	C_0
C_0:	**0**	1	0	0	0	0	1
C_1:	**1**	2	0	0	0	1	0
	1	3	0	0	0	1	1
C_2:	**0**	4	0	0	1	0	0
	0	5	0	0	1	0	1
	1	6	0	0	1	1	0
	1	7	0	0	1	1	1
C_3:	**0**	8	0	1	0	0	0
	0	9	0	1	0	0	1
	1	10	0	1	0	1	0
	0	11	0	1	0	1	1
	1	12	0	1	1	0	0
	1	13	0	1	1	0	1
	0	14	0	1	1	1	0
	1	15	0	1	1	1	1
C_4:	**1**	16	1	0	0	0	0
	0	17	1	0	0	0	1
	1	18	1	0	0	1	0
	1	19	1	0	0	1	1
	0	20	1	0	1	0	0
	1	21	1	0	1	0	1

Kao i ranije, kontrolni bit C_i određujemo tako da broj jedinica na pozicijama koje kontroliše taj bit u kodnoj reči (kolona **C**) bude paran. Na primer, za bit C_4 posmatramo pozicije koje u koloni C_4 imaju jedinice (to su pozicije od 16 do 21). Kako su bitovi poruke P na pozicijama od 17 do 21 redom 0, 1, 1, 0, 1, kontrolni bit C_4 postavljamo na 1, kako bi ukupan broj jedinica na pozicijama koje kontroliše bit C_4 bio paran.

Ukoliko se prilikom prenosa pojavi greška na poziciji 7 (tj. dobijemo kodnu reč $C' = 01100100010110110 1$) imaćemo sledeću tabelu:

	C	**Pozicija**	C_4	C_3	C_2	C_1	C_0
$C'_0 = 1$:	0	1	0	0	0	0	1
$C'_1 = 1$:	1	2	0	0	0	1	0
	1	3	0	0	0	1	1
$C'_2 = 1$:	0	4	0	0	1	0	0
	0	5	0	0	1	0	1
	1	6	0	0	1	1	0
Greška:	**0**	7	0	0	1	1	1
$C'_3 = 0$:	0	8	0	1	0	0	0
	0	9	0	1	0	0	1
	1	10	0	1	0	1	0
	0	11	0	1	0	1	1
	1	12	0	1	1	0	0
	1	13	0	1	1	0	1
	0	14	0	1	1	1	0
	1	15	0	1	1	1	1
$C'_4 = 0$:	1	16	1	0	0	0	0
	0	17	1	0	0	0	1
	1	18	1	0	0	1	0
	1	19	1	0	0	1	1
	0	20	1	0	1	0	0
	1	21	1	0	1	0	1

Odavde se lako zaključuje da je pozicija greške $C'_4C'_3C'_2C'_1C'_0 = 00111_{(2)} = 7$.

Na kraju ovog odeljka, prodiskutujmo efikasnost Hamingovog koda, u smislu udela korisnih bitova (bitova koji kodiraju korisnu informaciju) u kodnoj reči. *Koeficient efikasnosti* koda definišemo kao $c = m/n$, pri čemu je m broj bitova poruke koja se kodira, a n ukupan broj bitova kodne reči. Jasno je da je c uvek manje od 1, a poželjno je da bude što bliže jedinici (kako bi se smanjio udeo redundantnih, kontrolnih bitova). Odredimo ovaj koeficient za Hamingov kod. Naime, setimo se relacije $m + r + 1 \leq 2^r$. Za fiksirano r, najveći udeo korisnih bitova dobijamo tako što razmatramo poruku najveće moguće dužine za koju je dovoljno r kontrolnih bitova. Drugim rečima, najbolja efikasnost se postiže ako važi $m+r+1 = 2^r$. Otuda je $m = 2^r - r - 1$, a $n = m + r = 2^r - 1$. Sada je $c = m/n = (2^r - r - 1)/(2^r - 1) = 1 - r/(2^r - 1)$.

Odavde sledi da efikasnost raste sa r. Ovo je u skladu sa ranijom konstatacijom da broj potrebnih kontrolnih bitova r raste znatno sporije od dužine poruke m, te je bilo i logično očekivati da ćemo za duže kodne reči imati manji udeo kontrolnih bitova, a veći udeo korisnih bitova. Međutim, implementacija postupka određivanja kontrolnih bitova u slučaju poruka velike dužine postaje znatno komplikovanija. Hamingov kod u praksi obično nije moguće primeniti na celu poruku, iako bi se time teorijski postigla veća efikasnost, jer poruka može biti veoma dugačka i sadržati milione ili milijarde bitova. Zbog toga se Hamingov kod obično realizuje po blokovima: poruka se deli na blokove uzastopnih bitova fiksne dužine m, a zatim se za svaki od blokova zasebno određuje Hamingov kod (n, m) i dobija odgovarajući niz kodnih reči dužine n. Na ovaj način se dobija fiksiran koeficient efikasnosti $c = m/n$ koji se ne povećava sa dužinom poruke, ali se ni složenost implementacije ne povećava.

Otkrivanje grešaka obima većeg od 1

U ovom odeljku prikazujemo jedan postupak za otkrivanje nekih tipova grešaka čiji je obim veći od 1. U pitanju je takozvani *CRC* kod (engl. *cyclic redundancy check*). Napomenimo da ovaj postupak neće moći da otkrije sve greške nekog obima d, već se fokusira na određene tipove grešaka. Na primer, *lokalizovane (nagomilane) greške reda k* (engl. *burst errors*) su greške kod kojih su svi pogrešno očitani bitovi lokalizovani u nekom delu poruke čija je dužina k. Otkrivanje ovakvih grešaka ima veliku primenu u praksi.

CRC kod je zasnovan na deljenju polinoma nad poljem ostataka po modulu 2. Ovo polje se obično obeležava sa $\mathbb{Z}_2$, a odgovarajući prsten polinoma sa $\mathbb{Z}_2[X]$. Elementi polja $\mathbb{Z}_2$ su 0 i 1 (mogući ostatci pri deljenju sa 2), a operacije sabiranja i množenja date su sledećim tablicama:

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Čitalac neka samostalno za vežbu proveri da ovako definisane operacije zadovoljavaju aksiome polja. Ukoliko zamislimo da su u pitanju operacije nad bitovima, jasno je da sabiranje odgovara ekskluzivnoj disjunkciji (XOR), dok množenje odgovara konjunkciji (AND). Ovo omogućava efikasnu implementaciju ovih operacija u računaru. Primetimo da s obzirom da je $0 + 0 = 0$ i $1 + 1 = 0$ (tj. važi $(\forall x)(x + x = 0)$), zaključujemo da je svaki element samom sebi suprotan, tj. važi $(\forall x)(x = -x)$. Ovo znači da je $x - y := x + (-y) = x + y$, tj. oduzimanje je isto što i sabiranje.

Svojstva polja $\mathbb{Z}_2$ se prenose i na prsten polinoma $\mathbb{Z}_2[X]$. Koeficienti ovih polinoma su uvek 0 ili 1 (tj. svi monomi koji se pojavljuju u nekom polinomu su oblika x^i). Takođe, važi $P(x) = -P(x)$ za svaki polinom $P(x)$,

kao i $P(x) - Q(x) = P(x) + Q(x)$. Deljenje polinoma se izvodi na isti način kao i kod polinoma sa realnim koeficientima, s tim što je sada postupak jednostavniji, zbog toga što su koeficienti samo nule i jedinice. Na primer, posmatrajmo deljenje polinoma $P(x) = x^5+x^3+x^2$ polinomom $C(x) = x+1$:

$$
\begin{array}{rlllll}
 & x^5 & +x^3 & +x^2 & : \; x+1 = & x^4+x^3+x+1 \\
- & x^5 & +x^4 & & & \\
\hline
 & & x^4 & +x^3 & +x^2 & \\
- & & x^4 & +x^3 & & \\
\hline
 & & & & x^2 & \\
- & & & & x^2 & +x \\
\hline
 & & & & x & \\
- & & & & x & +1 \\
\hline
 & & & & 1 &
\end{array}
$$

Dakle, dobijamo količnik $Q(x) = x^4 + x^3 + x + 1$ i ostatak $R(x) = 1$. Primetimo da, ukoliko prilikom oduzimanja dobijemo minus ispred nekog monoma, taj minus možemo zameniti plusom, zato što je svaki element u prstenu $\mathbb{Z}_2[X]$ jednak svom suprotnom elementu. Npr, kada smo oduzimali x^2 i $x^2 + x$, razlika je, naravno, $-x$, ali smo mi pisali x (jer je $x = -x$). Operacija oduzimanja se efikasno može implementirati u računaru, pomoću ekskluzivne disjunkcije. Ukoliko svaki polinom predstavimo nizom bitova koji predstavljaju njegove koeficiente, tada se deljenje može implementirati tako što se delilac u svakom koraku poravna ispod odgovarajućih bitova deljenika, a zatim se na tim pozicijama izvrši operacija ekskluzivne disjunkcije.

Inače, polinom $x + 1$ je sam za sebe interesantan: ostatak pri deljenju ovim polinomom je uvek 0 ili 1 (jer pri deljenju polinomom stepena 1 možemo kao ostatak dobiti polinom stepena najviše 0, tj. element polja $\mathbb{Z}_2$). Takođe, može se pokazati da će polinom $P(x)$ biti deljiv polinomom $x+1$ (tj. ostatak će biti 0) ako i samo ako ima paran broj monoma. Polinomi sa neparnim brojem monoma daće ostatak 1 pri deljenju sa $x + 1$.

Vratimo se sada na CRC kod. Neka je data poruka P dužine m. Ovoj poruci pridružujemo polinom $P(x)$ iz $\mathbb{Z}_2[X]$ čiji su koeficienti upravo bitovi poruke P. Stepen ovog polinoma može biti najviše $m - 1$, ukoliko je vodeći bit poruke jedinica. Na primer, za poruku $P = 011010101$ dužine $m = 9$ imaćemo sledeći polinom $P(x) = 0 \cdot x^8 + 1 \cdot x^7 + 1 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x + 1$ (polinom stepena 7). Pretpostavimo da imamo neki, unapred zadat, polinom $G(x)$ stepena r (ovaj polinom nazivamo *generatorni polinom*). Postupak određivanja CRC koda poruke P svodi se na deljenje polinoma $P(x) \cdot x^r$ sa $G(x)$ (primetimo da je množenje polinoma $P(x)$ sa x^r ekvivalentno dopisivanju r nula na poruku P). Pretpostavimo da smo ovim deljenjem dobili količnik $Q(x)$ i ostatak $R(x)$. Količnik $Q(x)$ odbacujemo, jer nam nije potreban (u praksi se u implementaciji CRC algoritma količnik

uopšte i ne izračunava, već se određuje samo ostatak). Ostatak $R(x)$ će biti stepena najviše $r-1$, tj. imaće r koeficienata, pa ga možemo zapisati pomoću niske od r bitova. Kodna reč $C = \mathcal{K}(P)$ pridružena poruci P se dobija tako što se ovih r bitova dopišu na kraj poruke P kao kontrolni bitovi. Primetimo da ovako dobijenoj kodnoj reči C odgovara polinom $C(x)$ čiji su koeficienti bitovi kodne reči, a koji se može predstaviti u obliku $C(x) = P(x) \cdot x^r + R(x)$ (dopišemo r nula na poruku, a onda tih r nula zamenimo sa r bitova ostatka). Na primer, ako pretpostavimo da je generatorni polinom $G(x) = x^3 + x + 1$ (predstavljen niskom bitova 1011), tada deljenjem polinoma $P(x) \cdot x^3 = x^{10} + x^9 + x^7 + x^5 + x^3$ (koji odgovara prethodnoj poruci $P = 011010101$ dužine 9 na koju su dopisane 3 nule) polinomom $G(x)$ dobijamo ostatak $R(x) = x(= 0 \cdot x^2 + 1 \cdot x + 0 \cdot x^0)$, tj. kontrolne bitove 010. Sada je kodna reč $C = 011010101\mathbf{010}$.

Prilikom očitavanja sprovodi se sličan postupak, tj. očitana kodna reč C' se posmatra kao polinom $C'(x)$ koji se deli istim generatornim polinomom $G(x)$. Kako je $C(x) = P(x) \cdot x^r + R(x) = G(x) \cdot Q(x)$ (jer je $P(x) \cdot x^r = G(x) \cdot Q(x) + R(x)$, a važi $R(x) = -R(x)$), sledi da polinom G mora da deli polinom C. Otuda se kod CRC koda ispravnim kodnim rečima smatraju one kodne reči koje su *deljive generatornim polinomom*. Ukoliko deljenjem polinoma $C'(x)$ sa $G(x)$ utvrdimo da ostatak nije 0, tada smatramo da je dobijena kodna reč neispravna i prijavljujemo grešku.

Koje će greške biti moguće otkriti ovim postupkom zavisi od izbora polinoma $G(x)$. Naime, neka je $E(x) = C(x) - C'(x)$. Ovaj polinom se naziva *polinom greške*. Niz bitova koji odgovara ovom polinomu je upravo niz bitova koji bi se dobio primenom ekskluzivne bitovske disjunkcije nad kodnim rečima C i C' (tj. u tom nizu jedinice bi bile na onim pozicijama na kojima se desila greška). CRC postupak će otkriti grešku u očitanoj kodnoj reči C' ako i samo ako $G \nmid C'$, tj. ako i samo ako $G \nmid (C - C')$ (s obzirom da znamo da $G \mid C$), odnosno $G \nmid E$. Na primer, polinom $G(x) = x + 1$ omogućava otkrivanje svih grešaka neparnog obima (jer će ostatak pri deljenju $E(x)$ sa $x + 1$ biti 1 ako i samo ako $E(x)$ ima neparan broj monoma, tj. ako i samo ako imamo neparan broj grešaka). Otuda je korišćenje polinoma $x + 1$ kao generatornog polinoma ekvivalentno sa prostom kontrolom parnosti (zaista, imaćemo samo jedan kontrolni bit koji obezbeđuje da kodna reč bude deljiva sa $x+1$, tj. da imamo paran broj jedinica u njenom zapisu).

Sa druge strane, pretpostavimo da je polinom $G(x)$ stepena k, pri čemu $G(x)$ nije deljiv sa x^i ni za jedno $i > 0$ (ovo znači da mu je slobodan član jednak 1). Tada će ovakav polinom omogućiti otkrivanje svih lokalizovanih grešaka reda k. Zaista, ukoliko imamo lokalizovanu grešku reda k, tada se polinom greške $E(x)$ može zapisati kao $E(x) = E_1(x) \cdot x^p$, pri čemu je polinom $E_1(x)$ stepena manjeg od k (tj. niz bitova $E = C \oplus C'$ sadrži najpre određeni broj vodećih nula, zatim deo E_1 dužine k u kome su lokalizovane sve jedinice, i na kraju „rep“ dužine p koji se sastoji samo od pratećih nula). Kako je polinom $G(x)$ uzajamno prost sa x^p, to iz $G \mid E$ sledi da $G \mid E_1$,

što je moguće samo ako je stepen polinoma E_1 veći ili jednak od k, suprotno pretpostavci. Otuda je $G \nmid E$, tj. sve lokalizovane greške reda k mogu se otkriti pomoću ovakvog generatornog polinoma.