Felrättande koder

Thomas Höglund

KTH, Stockholm

Inledning. Felrättande koder används i CD-skivspelare, vid bildöverföring från satellit till jorden och vid kommunikationsradiotrafik – för att ta några exempel.

I samtliga dessa fall är det fråga om att överföra en bitföljd (d.v.s. en följd av nollor och ettor) från en sändare till en mottagare. Problemet är att följden kan bli störd under överföringen och att det mottagna meddelandet därför kan få dålig kvalitet.

Denna specialuppgift går att genomföra för hand men det är en fördel om du har tillgång till en dator då du kodar och avkodar meddelanden och då du simulerar störningar.

1. Tag ett pennset med 8 av våra vanligaste färger. Identifiera varje färg med en bitföljd av längd 3. T.ex.

rött	orange	gult	grönt	blått	violett	svart	vitt
000	001	010	011	100	101	110	111

Välj en enkel och tydlig färgbild som endast innehåller de färger du valt - en flagga t.ex. Lägg ett rutnät över bilden. Läs av färgen på rutorna i någon viss ordning. Du har nu fått en bitföljd som svarar mot din bild.

En annan möjlighet är att ta några meningar ur en vanlig text och översätta meningen till bitar t.ex. genom identifikationen: A = 00000, B = 00001, ..., Ö = 11011, mellanslag = 11100, punkt = 11101, komma = 11110, övrigt = 11111.

2. För att få en bild av vad störningar kan ställa till med ska du här simulera störningar. Gör ett kast med två tärningar för varje

bit i följden. Ändra biten (nolla till etta, etta till nolla) om båda tärningarna visar sexor, annars låter du biten stå kvar. I medeltal kommer alltså var trettiosjätte bit att ändras. Rita upp den mottagna bilden.

Allmänt om koder. Ett sätt att upptäcka fel är att sända varje bit 2 gånger, 0 kodas till 00 och 1 till 11. Om den mottagna följden är 10 så ser vi att ett fel uppstått men vi kan inte avgöra om den ursprungliga biten var 1 eller 0. Denna kod är alltså felupptäckande men inte felrättande. För att få en felrättande kod kan vi sända varje bit 3 gånger. Om den mottagna följden är 010 så ser vi att (minst) ett fel uppstått och att den ursprungliga biten (troligen) var 0. Risken finns naturligvis att två fel uppstått men denna risk är liten jämfört med sannolikheten att *ett* fel uppstått - förutsatt att felsannolikheten är liten och att fel uppstår oberoende av varandra. Denna kod rättar 1 fel på 3 bitar. En nackdel är emellertid att meddelandet förlängs med en faktor 3.

Mer allmänt kan vi dela in en bitföljd i block om k bitar, där k är ett positivt heltal, t.ex. $k = 2$. Låt n vara ett heltal som är större än k, t.ex. $n = 5$. Tilldela varje bitföljd av längd k en bitföljd av längd n. De senare bitföljderna kallas kodord och vi säger att vi valt en $[n, k]$ kod.

Om vi t.ex. har valt $[5, 2]$ koden:

$$00 \to 00000, \quad 01 \to 01011, \quad 10 \to 10101, \quad 11 \to 11110,$$

så kodas bitföljden 011011 så här:

$$011011 = 01 \;\; 10 \;\; 11 \to 01011 \;\; 10101 \;\; 11110.$$

Observera att endast ett fåtal av alla n-bitsföljder är kodord.

Avståndet mellan två bitföljder av längd n är antalet av de n positionerna där de två följderna har olika bitar. Så t.ex. är avståndet mellan 10101 och 11110 lika med 3 eftersom de skiljer sig på positionerna 2, 4 och 5.

Då en bitföljd sänts iväg och mottagits ska den avkodas. Om en mottagen n-bitsföljd inte är ett kodord så ersätts den med det närmaste kodordet - om ett sådant finnes. I exemplet ersätts följden 01000 med 00000 eftersom avståndet mellan dessa är 1 medan avståndet mellan 01000 och varje annat kodord är minst 2. Däremot kan vi inte avkoda följden 01100 eftersom denna har avståndet 2 till både 00000 och 11110 och avståndet 3 till de två övriga kodorden.

Denna kod rättar alltså ett fel och den är optimal i den meningen att det inte finns någon $[5, 2]$ kod som rättar fler fel. Problemet att för godtyckliga tal n och k finna en optimal $[n, k]$ kod är fortfarande (1988) olöst.

Vi ska nu införa en addition mellan bitföljder. Bitar adderas så här: $0+0=0$, $0+1=1+0=1$, $1+1=0$. Två lika långa bitföljder adderas genom att addera bitarna positionsvis. T.ex.

$$\begin{aligned}(1,1,1,0,0)+(0,1,0,1,0) &= (1+0,1+1,1+0,0+1,0+0)\\ &= (1,0,1,1,0).\end{aligned}$$

Här satte vi för tydlighetens skull parenteser runt bitföljderna och kommatecken mellan bitarna.

Hammingkoden. Denna kod är definierad för vissa n och k nämligen då $n = 2^r - 1$ och $k = n - r$ för $r = 2, 3, 4, ...$ eller $[n, k] = [3, 1]$, $[7, 4]$, $[15, 11]$, $[31, 26]$, ... Det är önskvärt att kvoten $\frac{n}{k}$ är nära 1 eftersom det kodade meddelandet förlängs med denna faktor. För Hammingkoden är dessa kvoter 3, 1.75, 1.36, 1.19, ... Däremot så rättar denna kod bara 1 fel i varje kodord av längd n.

Hammingkoden $[n, k]$ kan konstrueras på följande sätt:
– Skriv upp alla 2^r bitföljder av längd r. Stryk nollföljden och de r följder som innehåller exakt en etta. Kvar blir $2^r - 1 - r = k$ följder av längd r. I fallet $[7, 4]$ (d.v.s. $r = 3$) får vi de 8-1-3=4 följderna 011 101 110 111.
– Följden 10...0 av längd k kodas till det kodord av längd n vars k första bitar är just 10...0 och vars återstående r bitar är den första av de uppskrivna följderna. Kodordet svarande mot 010...0 fås genom att lägga till den andra av de kvarvarande följderna till 010...0 o.s.v. tills slutligen 0...01 kodas genom att lägga till den sista följden.

I exemplet
1000 → 1000011 0100 → 0100101 0010 → 0010110 0001 → 0001111.
– Koden är linjär: Om x kodas till $\bar{x}$ och y till $\bar{y}$ så kodas $x + y$ till $\bar{x} + \bar{y}$. T.ex. kodas 1010 till 1010101 eftersom 1010 = 1000 + 0010 och 1000011 + 0010110 = 1010101. Vidare kodas 1011 till 1011010 eftersom 1011 = 1010 + 0001 och 1010101 + 0001111 = 1011010 o.s.v.

3. Låt för varje kodord x, $B(x)$ beteckna mängden av alla bitföljder av längd n som har avståndet högst 1 till x. Övertyga dig om att följande resonemang är riktigt i första hand för $r = 2$ och $r = 3$ men även för ett allmänt r. Avståndet mellan två olika kodord är alltid minst 3. $B(x)$ består av $n + 1 = 2^r$ kodord. Om x och y är två olika kodord så finns ingen bitföljd av längd n som ligger i både $B(x)$ och $B(y)$. Det finns $2^k = 2^{n-r}$ kodord, så de 2^{n-r} mängderna $B(x)$ innehåller därför $2^{n-r} \times 2^r = 2^n$ olika bitföljder av längd n. Därför finns ingen bitföljd av längd n som ligger utanför alla $B(x)$.

Vi har alltså visat att för varje bitföljd av längd n finns exakt ett kodord på avståndet högst 1.

4. Koda bitföljden under punkt 1 med Hammingkoden $[7, 4]$. Simulera störningar som under punkt 2. Avkoda. Rita bilden. Jämför med resultatet under punkt 2.

5. I de i inledningen nämnda exemplen kommer störningarna oftast i skurar (flera fel i följd). För att simulera en sådan situation kan du göra så här: Kasta två tärningar. Ändra alla de tre första bitarna om du får två sexor, annars låter du bitarna stå kvar. Upprepa för de tre följande bitarna o.s.v. Här kommer alltså störningarna tre och tre men vi har fortfarande att i medeltal är var trettiosjätte bit störd. Eller hur? Avkoda och jämför med tidigare resultat.

Multiplikation av bitföljder. Vi har tidigare definierat addition av bitföljder. För att konstruera koder som klarar av störningar av den typ du simulerade under punkt 5 ska vi nu även införa en multiplikation.

Varje bitföljd $(b_0, b_1, ..., b_{m-1})$ svarar mot ett polynom $b_0 + b_1x + b_2x^2 + ...b_{m-1}x^{m-1}$ och omvänt. Så t.ex. svarar 101 mot $1 + 0 \cdot x + 1 \cdot x^2 = 1 + x^2$ och 010 mot $0 + 1 \cdot x + 0 \cdot x^2 = x$. Följden 000 svarar mot nollpolynomet (talet 0) och 100 mot det polynom som är konstant 1. Det finns två polynom av grad 1: x och $1 + x$. Det finns fyra polynom av grad 2: x^2, $x + x^2$, $1 + x^2$ och $1 + x + x^2$. De tre första av andragradspolynomen går att skriva som en produkt av två polynom av lägre grad, nämligen $x^2 = x \cdot x$, $x + x^2 = x(1 + x)$ och $1 + x^2 = (1 + x)(1 + x)$. (Här utnyttjade vi att $1 + 1 = 0$.) Polynomet $1 + x + x^2$ går däremot inte att skriva som en produkt av två polynom av lägre grad. Sådana polynom kallas irreducibla (jämför primtal). Det finns alltså ett irreducibelt polynom av grad 2: $1 + x + x^2$.

Betrakta nu bitföljder av längd 2. Det finns 4 sådana: 00, 10, 01 och 11 och de svarar mot polynomen 0, 1, x och $1 + x$. När vi multiplicerar dessa ska vi räkna som om $1 + x + x^2 = 0$ d.v.s. som

om $x^2 = 1+x$. (Addera $1+x$ till båda sidor och utnyttja att $2 = 0$.) Vi får $x \cdot x = x^2 = 1 + x$, $x(1 + x) = x + x^2 = x + (1 + x) = 1$, $(1 + x)(1 + x) = 1 + 2x + x^2 = 1 + x^2 = 1 + (1 + x) = x$.

För att slippa skriva så mycket skriver vi $0, 1, 2, 3$ i stället för 00, 10, 01 respektive 11. D.v.s. i stället för $0, 1, x$ respektive $1 + x$. Vi har alltså multiplikationstabellen

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

Till höger står den additionstabell den addition vi redan infört ger.

Observera att vi kan räkna med dessa element ungefär som med vanliga tal. Vi kan subtrahera: För varje p och q finns exakt ett r så att $r + p = q$ (nämligen $r = p + q$). Vi kan även dividera med p om $p \neq 0$: För varje q finns exakt ett s så att $sp = q$. Om vi skriver q/p i stället för s så har vi t.ex. $3/2{=}2$ och $1/2{=}3$. Lägg också märke till att alla element, p, uppfyller $p^4 = p$.

Det finns två irreducibla tredjegradspolynom: $1 + x + x^3$ och $1 + x^2 + x^3$. Välj t.ex. det första. Räkna som om detta är 0 (d.v.s. ersätt x^3 med $1 + x$) när du multiplicerar polynom av grad högst 2. Det medför att x^4 ska ersättas med $x + x^2$ eftersom $x^4 = x \cdot x^3 = x(1 + x) = x + x^2$. Så till exempel är $7 \cdot 5 = 6$ eftersom

$$\begin{aligned}(1 + x + x^2)(1 + x^2) &= 1 + x + 2x^2 + x^3 + x^4 \\ &= 1 + x + 0 + (1 + x) + (x + x^2) = x + x^2.\end{aligned}$$

6. Gör en multiplikationstabell och en additionstabell för de 8 bitföljderna av längd 3. Gör även en tabell över $p, p^2, p^3, p^4, p^5, p^6, p^7$

för alla de 8 elementen p. Även här kan vi subtrahera och dividera. Observera att alla element p, uppfyller $p^8 = p$ och att vissa element är sådana att de 7 potenserna p^i, $i = 1, ..., 7$ ger alla element utom 0. Sådana element kallas primitiva. Vilka element är primitiva? Om p är primitivt så gäller $1 + p + p^2 + ... + p^6 = 0$.

Reed-Solomon koden. På motsvarande sätt kan man definiera multiplikation för bitföljder av längd m. Reed-Solomon koden kodar bitföljder av längd mk bitar till bitföljder av längd $m(2^m - 1)$ bitar. Här är m och k är positiva heltal.

Vi ska i fortsättningen hålla oss till fallet $m = 3$, $k = 5$. De bitföljder som ska kodas har alltså längden $3 \times 5 = 15$ bitar och kodorden har längden $3(2^3 - 1) = 3 \times 7 = 21$ bitar (5 element kodas till 7 element). Meddelandet förlängs alltså med en faktor 1.4 att jämföra med 1.75 för Hammingkoden [7,4].

Reed-Solomon koden kan konstrueras så här: Välj ett primitivt element p, t.ex. $p = 2 = (0, 1, 0)$. Dela in 15-bitsföljden i 5 block av längd 3. Vi får då en följd $(a_0, a_1, a_2, a_3, a_4)$ av element. Denna kodas till de 7 elementen $(c_0, c_1, ..., c_6)$, där

$$(\star) \qquad c_i = a_0 + a_1 p^i + a_2 p^{2i} + a_3 p^{3i} + a_4 p^{4i}$$

för $i = 0, 1, 2, 3, 4, 5, 6$. Här har vi vanlig multiplikation i exponenterna $2i$, $3i$ och $4i$.

Så t.ex. om vi startar med 15-bitsföljden 100010000110111 och låter $p = 2$ så är $(a_0, a_1, a_2, a_3, a_4) = (1, 2, 0, 3, 7)$ och vi får

$$c_i = 1 + 2 \cdot 2^i + 0 \cdot 2^{2i} + 3 \cdot 2^{3i} + 7 \cdot 2^{4i}.$$

Därför är

$$\begin{aligned} c_0 &= 1 + 2 + 0 + 3 + 7 = 7, \\ c_{16} &= 1 + 2 \cdot 2 + 0 \cdot 2^2 + 3 \cdot 2^3 + 7 \cdot 2^4 \\ &= 1 + 4 + 0 + 5 + 4 = 4 \end{aligned}$$

o.s.v.

Addera de 7 identiteterna ($\star$) och använd dig av att $1+p+p^2+...+p^6=0$. Resultatet blir

$$c_0+...+c_6=7a_0+0\cdot a_1+...+0\cdot a_4=a_0.$$

Att $7a_0=a_0$ kommer sig av att $a_0+a_0=0$. Låt q vara sådant att $qp=1$ d.v.s. $q=p^6$ ($q=5$ med vårt val av p). Multiplicera båda sidorna i ($\star$) med q^i och addera. Resultatet blir

$$c_0+c_1q+c_2q^2+...+c_6q^6=a_1.$$

(Kom ihåg att $p^8=p, p^9=p^2, ..., p^{15}=p, ...$) Multiplicera sen ($\star$) med q^{2i} och addera. Fortsätt så med q^{3i},...,q^{6i}. Resultatet blir ett avkodningsrecept:

$$a_k=c_0+c_1q^k+c_2q^{2k}+c_3q^{3k}+c_4q^{4k}+c_5q^{5k}+c_6q^{6k}$$

för $k=0,1,2,3,4$. Dessutom får vi två ekvationer som gäller för kodord (men inga andra):

($\star\star$) $$c_0+c_1q^k+c_2q^{2k}+c_3q^{3k}+c_4q^{4k}+c_5q^{5k}+c_6q^{6k}=0$$

för $k=5,6$. Dessa ekvationer kan användas till att ge en algoritm som rättar ett fel men det behöver du inte göra.

7. Denna kod kan rätta fel i ett av de 7 elementen (och därmed 3 fel i de 3 motsvarande bitarna). Du ska nu övertyga dig om att detta påstående är sant genom att fylla i detaljerna i följande resonemang.
– Om $c'=(c'_1,...,c'_6)$ och $c''=(c''_1,...,c''_6)$ är två kodord så är $c=c'-c''=(c'_1-c''_1,...,c'_6-c''_6)$ också ett kodord.
– Det räcker därför att visa att om $c=(c_1,...,c_6)$ är ett kodord sådant att $c_i=0$ för alla utom möjligen två värden på i så är $c_i=0$

för alla i.
– Antag att $c_i = 0$ för alla utom möjligen två värden på i ($i = 0$ och $i = 1$ t.ex.). Eftersom de två ekvationerna ($\star\star$) är uppfyllda så måste även c_0 och c_1 vara 0.

8. Koda din bild med denna kod. Simulera störningar (i bitföljden) som under punkt 5. Avkoda och rita upp bilden. Du behöver inte använda avkodningsreceptet. Eftersom du vet vad rätt svar ska vara så behöver du i regel bara kontrollera att den störda följden $(\tilde{c}_0, ..., \tilde{c}_6)$ skiljer sig från det ostörda kodordet $(c_0, ..., c_6)$ på högst en av de 7 positionerna. Om så inte är fallet kan du nöja dig med att konstatera att fel uppstått.

Litteratur

MacWilliams, F.J. and Sloane, N.J.A., *The theory of error-correcting codes*. North-Holland 1977.