Föreläsning 5

MA1508

Att förebygga mänskliga fel

Jag beskrev en metod som kallas mod 37 metoden förra gången.

En följd $(a_1, a_2, \ldots, a_n)$ är ett kodord om

$$1 \cdot a_1 + 2 \cdot a_2 + \ldots n \cdot a_n = 0 \pmod{37}.$$

Exempel 1. Är $(0, 0, 7, 4)$ ett tillåtet kodord? Ja, för att

$$1 \cdot 0 + 2 \cdot 03 \cdot 7 + 4 \cdot 4 = 21 + 16 = 37 \equiv 0 \pmod{37}.$$

△

Om Alice skriver fel i en siffra eller byter plats på två intilliggande siffror kommer den koden upptäcka det. Det visar sig att det är viktigt för den egenskapen att 37 är ett primtal.

I boken räknar de åt andra hållet, så det är sista "siffran" som multipliceras med 1 osv. Det påverkar inget viktigt, kodorden med min definition blir omvändningen av bokens kodord.

Denna mod 37 metod är tänkt att användas när man matar in alfanumeriska koder. Om vi ska bara mata in vanliga decimala siffror så finns en enklare metod som heter mod 11-metoden, som används i t ex ISBN-nummer. ISBN-nummer består av 10 siffror, men den sista siffran kan ibland vara ett X. Det står i så fall för 10. För att kontrollera om det är giltigt multiplicerar man sista siffran med 1, näst sista med 2, osv. Summera och om resultatet är lika med 0 modulo 11 så är numret giltigt.

Exempel 2. Låt oss använda en kortare variant av ISBN-nummer med bara 4 siffror. Är $104X$ ett giltigt nummer? Vi räknar $10 + 2 \cdot 4 + 3 \cdot 0 + 4 \cdot 1 = 22$. Det är delbart med 11 så numret är giltigt.

△

Exempel 3. Om ett fyrsiffrigt ISBN-nummer börjar med 337, vad måste sista siffran vara? Den måste vara 9. △

Precis som mod 37 metoden kommer ISBN-algoritmen upptäcka alla fel i enstaka siffror, och alla omkastningar av två grannsiffror. Varför upptäcker den omkastningar kanske ni undrar? Låt oss t ex anta att vi kastar om siffrorna nästa sista och tredje sista siffran från slutet. Antag att det fortfarande skulle gälla att hela den viktade summan blir delbar med 11. Om vi kallar de två berörda siffrorna för x och y måste det då gälla att $3x+2y \equiv 3y+2x \pmod{11}$. Det är ekvivalent med att $x \equiv y \pmod{11}$. Men då måste $x = y$, eftersom x och y är entalssiffror. Att kasta om två siffror som är samma är givetvis inte ett fel.

Varför upptäcker den fel i en enstaka siffra? Jo, låt oss anta felet är på plats k från slutet, och att vi byter ut x mot y. Om $k \cdot x \equiv k \cdot y \pmod{11}$ så måste $k(x-y)$ vara delbart med 11. Men så länge k är ett tal mellan 1 och 10 betyder det att $x - y$ är delbart med 11. Eftersom x och y är mellan 0 och 10 så betyder det att $x = y$.

Att 11 är ett primtal är viktigt här. Om k vara en delare till 11 så kunde $k(x-y)$ bli delbart med 11 utan att $x = y$. Det är därför man låter sista siffran vara X och räknar modulo 11, trots att man kanske egentligen önskade att jobba med decimala siffror.

Ett alternativ om man vill bara jobba med decimala siffror är Luhns algoritm. Den handlar inlämningsuppgiften om till största delen. Den fungerar lite likt mod 11-metoden eller mod 37-metoden men det finns också skillnader som ni ser om ni läser uppgiften. Alla siffrorna är då mellan 0 och 9. Nackdelen är att den inte längre upptäcker riktigt alla utbyte av två grannsiffror, utan om det är 0 och 9 som byter plats så upptäcker den inte det. En annan nackdel kanske är att den kan kännas lite mer komplicerad?

En mer komplicerad algoritm som heter Verhoeffs algoritm upptäcker både fel i en enstaka siffra, och utbyte av två grannsiffror utan att använda andra siffror än 0-9. Nackdelen är att den inte är så enkel att beskriva.

Binär aritmetik

Vi går nu in i kapitel 3. Boken börjar med att repetera binär aritmetik. Det har vi redan repeterat. Viktigast för oss i detta avsnitt är kanske att lägga märke till att addition och subtraktion är samma sak när man räknar modulo 2.

Repetitionskod

Låt oss nu ge en kod som låter oss inte bara upptäcka fel, utan också korrigera (vissa av) dem.

Anta att kodlängden är en multipel av 3. Vi säger då att kodorden i vår repetitionskod består av de kodord där första tredjedelen är identisk med andra tredjedelen och med sista tredjedelen.

Exempel 4. Om kodlängden är 6 så är 010101 ett kodord, men 110101 är inte ett kodord.

Om Bob tar emot 110101 så kan han alltså veta att något blivit fel. Men nu kan han göra mer än så. Om han antar att fel i en bit är mycket sannolikare än fel i två bitar kan han göra gissningen att det nog var 010101 som Alice skickade.

△

Att kunna göra felrättning och inte bara upptäcka fel kan vara väldigt viktigt ibland. Tänk t ex att Bob läser data som är sparat på en hårddisk eller CD-skiva och det blivit fel på en av bitarna i det fysiska lagringsmediet. Om han bara upptäckte felet kan han inte gärna be Alice skicka meddelandet igen. (Betyder det att han ska köpa en ny CD-skiva?) Men om koden har felrättande egenskaper kan han göra en gissning för vad det borde stå och ändå använda datan.

Definition. En (n, k)-kod är en kod med längden n och 2^k kodord. *Takten* på koden sägs vara k/n. Takt heter rate på engelska.

Om vi har en (n, k)-kod innebär det att vi kan koda om k bitar till n bitar innan överföringen.

Vår tripla repetitionskod ovan har takten $1/3$.

Definition. En kod sägs vara t-felrättande kod om Bob kan hitta rätt kodord så länge han vet det skett t eller färre fel i överföringen. Den sägs vara en e-felupptäckande kod om e fel eller färre inte kan förvandla ett kodord till ett annat kodord.

Vår tripla repetitionskod är 1-felrättande och 2-felupptäckande.

Om en kod är e-felsupptäckande så kan Bob åtminstone se att det uppstått fel om de inte är fler än e stycken.

Låt oss återgå till vår tripla repititionskod av längd 6. Vi kan skriva upp ett antal ekvationer som avgör om $(x_1, x_2, x_3, x_4, x_5, x_6)$ är ett riktigt kodord. De riktiga kodorden är nämligen precis de för vilka följande ekvationssystem gäller modulo 2

$$\begin{cases} x_3 = x_1 \\ x_5 = x_1 \\ x_4 = x_2 \\ x_6 = x_2 \end{cases}.$$

Om vi vill kan vi skriva om det som

$$\begin{cases} x_1 + x_3 = 0 \\ x_1 + x_5 = 0 \\ x_2 + x_4 = 0 \\ x_2 + x_6 = 0 \end{cases}.$$

Matriser

En *matris* är en rektangulär uppsättning av tal.

Exempel 5. Här är en med reella tal:

$$\begin{pmatrix} \pi & 3 & 4 \\ 2 & 3 & -1 \end{pmatrix}$$

△

Exempel 6. Här är en annan matris med reella tal.

$$\begin{pmatrix} 4 & 2 & 3 \\ 1 & 2 & 9 \\ -2 & 2 & 3 \end{pmatrix}.$$

△

Vi kan ibland addera matriser. Har vi två matriser med samma dimensioner, dvs de har lika många rader båda två, och lika många kolonner båda två, då kan vi addera dem.

Exempel 7.

$$\begin{pmatrix} 3 & 2 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} -4 & 2 \\ 5 & 8 \end{pmatrix} = \begin{pmatrix} -1 & 4 \\ 6 & 9 \end{pmatrix}$$

△

Vi kan också ibland multiplicera matriser. Vi beskriver ett fall: Om A är en matris med n rader och m kolonner och B är en matris med m rader och 1 kolonn kan vi multiplicera A med B. (Det ska alltså vara lika många rader i B som det är kolonner i A.) Matrisen som vi får som produkt kommer ha lika många rader som A och 1 kolonn.

Vi beskriver multiplikationen i ett exempel

Exempel 8.

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 2 & -1 & 0 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \cdot 1 + 3 \cdot (-1) + 5 \cdot (-2) + 7 \cdot 3 \\ 2 \cdot 1 + (-1) \cdot (-1) + 0 \cdot (-2) + 3 \cdot 3 \end{pmatrix} = \begin{pmatrix} 9 \\ 12 \end{pmatrix}.$$

△