SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

QR-koder: Steg för steg

av

Alvin Malmqvist

2024 - No K1

QR-koder: Steg för steg

Alvin Malmqvist

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Per Alexandersson

2024

Abstract

In this thesis, we delve deep into the world of error-correcting codes, focusing on the mathematical structures that underpin our everyday communication. The study begins by rediscovering error-correcting codes in an intuitive and relatable way, illustrating how the communication methods we use daily have mathematical counterparts. This leads to a foundational discussion, which sets the stage for the more advanced theory covered later, on repetition codes and linear codes. Certain claims and theorems along the way naturally require proof, something this thesis does not shy away from providing.

Once we have built a solid understanding of linear codes, we proceed to introduce Reed-Solomon codes, a cornerstone of modern society. These codes are used in everything from CDs and DVDs to communication with the Voyager spacecraft. We start with a somewhat simplistic approach, quickly increasing the pace and complexity of the codes until we are finally equipped to tackle QR codes. In the final chapter of the thesis, everything we have learned is applied to create a QR code from scratch, step by step.

Sammanfattning

I den här uppsatsen dyker vi djupt ner i de felrättande kodernas värld, med fokus på de matematiska strukturerna som bygger upp vår vardagskommunikation. Studien inleds med att återupptäcka felrättande koder på ett sätt som är intuitivt och lätt att relatera till, för att sedan visa kommunikationsmetoder vi använder oss av dagligen har matematiska ekvivalenter. Detta leder oss till en diskussion, som kommer att ligga till grund för all den mer avancerade teorin vi går igenom sen, om repetitionskoder och linjära koder. Vissa påståenden och satser längs vägen kräver såklart bevis något som den här uppsatsen inte viker sig för att ge.

När vi väl byggt upp vårt kunskapsförråd om linjära koder är det dags att introducera Reed-Solomon-koder, något som används i hela det morderna samhället. Allt ifån CD, och DVD-skivor, till kommunikation med rymdsonden Voyager. Vi börjar nästan lite simplistiskt, för att raskt öka tempot och komplexiteten på koderna, tills vi sist är utrustade för att tackla QR-koder. I uppsatsens sista kapitel används allting vi lärt oss för att skapa en egen QR-kod, från grunden, steg för steg.

Innehåll

1 Introduktion **5**

2 Linjära koder **7**

2.1 Idén bakom självrättande koder 7
2.2 Viktiga definitioner och koncept för linjära koder 8
2.3 Repetitionskoder 9
2.4 Minsta Hammingavståndet för linjära koder 10

3 Reed-Solomon-koder **13**

3.1 Lagrange's metod för polynominterpolering 13
3.2 Lagrangeinterpolering steg för steg 14
3.3 Unikheten av Lagrange'spolynom 16
3.4 Att koda en Reed-Solomon-kod 16
3.5 Att avkoda en Reed-Solomon-kod 17
3.6 Exempel på kodning och avkodning av Reed-Solomon-koder 19
3.7 Systematiska Reed-Solomon-koder 19
3.8 Kodning av systematiska BCH-Reed-Solomon-koder 20
3.9 Avkodning av systematiska BCH-Reed-Solomon-koder 21
3.10 Exempel på systematiska BCH-Reed-Solomon-koder 25
3.11 Primitiva polynom för att generera $\mathbb{F}_{p^q}$ 26

4 QR-koder **29**

4.1 Att välja meddelande och QR-kodversion 30
4.2 Byte padding, och konstruktion av datasträng 31
4.3 Felkorrigering 32
4.4 Fasta mönster 42
4.5 Rita ut data 44
4.6 Att välja det bästa maskeringsmönstret 45
4.7 Rita in formatinformation 48

Referenser **51**

1 Introduktion

Den största skillnaden mellan människor och djur, om vi bortser från att människor onekligen är djur, är vår förmåga att kommunicera, att utbyta idéer och komplexa tankar sinsemellan. Den mänskliga kommunikationens historia började sannolikt med enklare former av läten och gester. Allteftersom de mänskliga samhällena växte sig större och mer avancerade ökade behovet av sofistikerade och mer strukturerade former av kommunikation. Först kom talade språk, som tillät oss att resonera med varandra, och att uttrycka känslor.

Därefter kom skriftspråket, och då blev det för första gången möjligt att kommunicera med personer genom tiden. Detta möjliggjorde att samhällen kunde stifta lagar, och bilda diplomatiska och politiska relationer med andra samhällen, men också att utveckla skilda kulturer och religioner. Några tusen år senare uppfanns tryckpressen i Europa vilket födde masskommunikation på det sätt vi är vana vid idag. Då började det istället att handla om att kommunicera så snabbt som möjligt. Stora framsteg gjordes på 18- och 1900-talet med uppfinningar som telegrafi, telefoni, och radio.

I det moderna samhället kommunicerar vi dagligen över stora avstånd tack vare internet, smartphones, och satellit-teknik. Sociala medier och instant-messaging har till stor del tagit över vår dagliga kommunikation. Men vartefter dessa digitala kommunikationsmedel ökade i popularitet, gjorde också behovet av att informationen bevaras, även i miljöer där brus förekommer. Detta ledde till utvecklandet av felrättande koder.

2 Linjära koder

2.1 Idén bakom självrättande koder

När man först hör talas om det faktum att det finns koder som på egen hand kan fixa och korrigera fel som uppstått så låter det nästan för fantastiskt för att vara sant. Hur kan en kod, som bara är en sträng symboler, möjligtvis veta att något är fel? Och till och med mer än så... rätta till de fel som har uppstått?

Kanske har du någon gång varit med om att en kompis varit ute en fredagskväll och messat dig något i stil med "hek ksn du hänta mig jah ät på tikardz lub.Meddelandet kan vara lite svårt att tyda, men om vi tillämpar lite kontext kanske det går att tyda ändå. Dels finns en lingvistisk kontext, till exempel är varken "hek"eller "jahörd som ingår i det svenska språket. Då kan vi anta att det blivit fel när vår käre vän träffat fel tangenter. Vi kan också tillämpa en social kontext. Om vi vet att hen varit ute och druckit, och puben kanske ligger en bit bort, så är det naturligt att behöva hjälp att ta sig hem. Om vi lägger ihop allt så kan vi istället tolka meddelandet som "hej kan du hämta mig jag är på Rikards pub,som är mer rimligt.

Den delen av kodningsteori som behandlar självrättande och felrättande koder, två begrepp som kommer att användas synonymt under denna uppsats, handlar om att försöka återskapa den kontext som finns när vi kommunicerar med varandra matematiskt.

Tänk dig att du försöker stå och prata med någon på en fest(som matematiker är det något jag endast hört talas om) där det spelas hög musik. Att prata är absolut möjligt, men det är svårt. Om din konversationspartner på något sätt indikerar att denne inte korrekt uppfattat vad du sagt, genom att till exempel utbrista "va?"eller helt enkelt inte svara på det du sagt. Då kommer de flesta rent instinktivt att repetera vad de sagt så att personen får en ny chans att tolka vad du sade.

Idén om att repetera sig kommer lite senare leda oss till att upptäcka repetitionskoder, en typ av linjär kod. Låt oss börja med att definiera terminologin kring linjära koder, så kan vi komma tillbaka till repetitionskoder när vi utvecklat vår förståelse lite grann.

2.2 Viktiga definitioner och koncept för linjära koder

Följande definitioner bygger på boken Discrete Mathematics av Norman L. Biggs. [Big02]

Definition 2.1. En kod är en mängd strängar. En kod betecknas ofta $\boldsymbol{C}$.

Definition 2.2. Om $c \in \boldsymbol{C}$ kallas c för ett kodord.

Exempel 2.3. $\boldsymbol{C}_1 = \{hej, på, dig\}$ är en kod. Strängen hej är ett kodord eftersom $hej \in C_1$.

Definition 2.4. En kod $\boldsymbol{C}$ kallas linjär om den är en delgrupp V^n, där V^n är vektorrummet som innehåller alla kodord av längd n.

Definition 2.5. I en linjär kod kommer alla kodord att vara lika långa. Vi kallar det för kodordets längd och det betecknas med n.

Definition 2.6. Även meddelandena som skickas kommer att behöva vara lika långa. Längden på ett meddelande kallas kodens dimension och betecknas k. Om en kod är definierad över ett vektorrum av dimension q kommer antalet kodord i koden att vara q^k.

Planen här är alltså att välja ut ett meddelande på k symboler, med hjälp av någon förutbestämd algoritm göra om detta till ett kodord med n symboler. Men varför det? Jo, det gör att vi skapar lite distans mellan orden. Då kan vi, ifall något fel skulle uppstå [1] och mottagaren ser att det kodord denne tar emot inte finns i vår kod, så kan hen leta upp det mottagna, felaktiga, kodordets närmaste granne i koden. Då måste vi hitta något sätt att definiera något sätt att hitta avståndet mellan två strängar.

Definition 2.7. Hammingavståndet mellan två strängar av samma längd, c_1 och c_2 betecknas $\delta(c_1, c_2)$ och är definierat som antalet platser där kodorden skiljer sig. Om en kod $\boldsymbol{C}$ uppfyller att, för alla kodord $c_x, c_y \in \boldsymbol{C}$, $\delta(c_x, c_y) \leq d$ kallar vi d för det minsta Hammingavståndet.

Exempel 2.8. Hamming avståndet δ(matematik,statistik) = 6. Båda strängarna har är lika långa(9 symboler), och de skiljer sig från varandra på 6 ställen.

[1] alltså att en symbol byts ut mot en annan

Definition 2.9. Om vi skickar ett kodord c till vår mottagare, som tar emot ordet w, säger vi att ett fel uppstått ifall $\delta(c, w) \neq 0$.. Vidare säger vi att det uppstått s fel ifall $s = \delta(c, w)$.

Definition 2.10. Om ett eller flera fel uppstår i vår kod korrigerar vi w till det kodord c som gör att $\delta(c, w)$ blir så liten som möjligt.

2.3 Repetitionskoder

Om vi går tillbaka till exemplet där du försöker prata med någon över hög musik, och det är uppenbart att personen inte förstått dig, så du repeterar det du sagt, så ska vi nu försöka återskapa det matematiskt.
Låt oss säga att kontexten till konversationen på den högljudda festen är att din kompis har ställt dig en fråga, till exempel "vill du gå hem?-– en fråga som kräver ett ja eller ett nej. Vi väljer att representera ett ja med en 1:a, och ett nej med en 0:a. Skulle det vara så att vi har jättekul på festen, och *inte* vill gå hem, svarar vi alltså 0. Men nu uppstår det ett fel! I den brusiga kanalen som meddelandet överförs genom byts vår 0:a ut mot en 1:a, så kompisen börjar packa sina saker och undrar sen varför du inte gjort samma sak. För att förhindra att sådana fel förstör vår kväll så måste vi hitta på en kod.

Definition 2.11. En linjär kod kallas för en repetitionskod ifall vi kodar ett meddelande m genom att repetera m ett antal gånger.

Exempel 2.12. En repetitionskod som liknar vårt exempel ovan hade kunnat vara ifall vi istället för att skicka en 1:a eller en 0:a, skickar 00 eller 11. Det ger oss att vår kod $\boldsymbol{C} = \{00, 11\}$.
Om vi skapar en sådan kod, skickar 11, och vår mottagare får ordet 01, så vet denne att något blivit fel eftersom $01 \notin \boldsymbol{C}$. Det finns bara ett problem. $\delta(01, 00) = \delta(01, 11) = 1$, så vi vet inte vad vi ska korrigera till. Vi kan se *att* det uppstått ett fel, men vi har inte tillräckligt med information för att fixa det. Vi kallar det för att vi kan detektera eller upptäcka ett fel.

Vi ändrar koden lite grann, vi repeterar inte meddelandet en gång längre, utan två gånger, alltså blir vår kod $\boldsymbol{C} = \{000, 111\}$. Om vi nu skulle skicka ett ja(111) och vår mottagare tar emot 110, så vet denna att svaret var just "ja"eftersom $\delta(110, 111) = 1 > \delta(110, 000) = 2$. Observera att ifall två fel uppstår, till exempel ifall det mottagna ordet skulle vara 010 skulle detta korrigeras till 000, alltså nej.När vi arbetar med felrättande koder utgår vi ifrån att fel är ovanliga.

2.4 Minsta Hammingavståndet för linjära koder

Sats 2.13. *Singletongränsen. För alla linjära koder av längd n och dimension q gäller att $d \leq n - k + 1$. [MS04]*

Bevis. Låt oss anta att vi har en linjär kod $\boldsymbol{C}$ med dimension k, längd n, och minsta Hammingavstånd d.

Vi fastslår nu de första $d-1$ symbolerna i koden, vilket ger oss att dessa positioner har q^{d-1} olika kombinationer. Eftersom vi fastslagit $d-1$ positioner måste det finnas $n-(d-1)$ positioner kvar, enligt definitionen av längden av ett kodord n.

Efter att ha fastslagit $d-1$ positioner måste fortfarande alla kodord i $\boldsymbol{C}$ vara entydiga, eftersom enligt definitionen av det minsta Hammingavståndet d krävs det att alla kodord skiljer sig på minst d positioner.

Vi har alltså q^k unika kodord i $\boldsymbol{C}$. Dessa kodord måste alltså, enligt Dirichlets lådprincip, få plats i de sista $n-(d-1)$ positionerna. Alltså gäller det att

$$q^k \leq q^{n-(d-1)} \implies k \leq n-(d-1) \implies d \leq n-k+1.$$

□

Med hjälp av Singletongränsen inser vi ganska lätt att, ifall det uppstår $d-1 \leq n-k$ fel, eller färre, så kommer vi att få ett resultat som inte är ett kodord i $\boldsymbol{C}$, och vi kan anta att något gått snett under överföringen. Det kallas att vi kan detektera upp till $d-1 = n-k$ fel, men vi pratar ju ändå om fel*rättande* koder, inte fel*upptäckande* koder. Hur många fel kan en kod med minsta Hammingavstånd d rätta till?

Sats 2.14. *En kod med minsta Hammingavstånd d kan korrigera upp till $\left\lfloor \frac{d-1}{2} \right\rfloor = \left\lfloor \frac{n-k}{2} \right\rfloor$ fel. Vi korrigerar fel genom att jämföra en mottagen sträng med alla kodord, och ta det kodord som har lägst Hammingavstånd.*

Bevis. Låt oss anta att vi har en kod med minsta Hammingavstånd d. Låt oss vidare anta att vi tar emot ett ord w som inte är ett kodord, och att det uppstått $s \leq \left\lfloor \frac{d-1}{2} \right\rfloor$. Till sist antar vi att vi inte kan bestämma vilket kodord w ska korrigeras till eftersom det gäller att $\delta(w,c) = \delta(w,c^*) = s$.

Med hjälp av triangelolikheten kan vi alltså skriva att

$$\delta(c,c^*) \leq \delta(w,c) + \delta(w,c^*) = s + s = 2s. \tag{1}$$

Om vi använder definitionen av Hammingavståndet, d.v.s. att $d \leq \delta(c, c^*)$, och stoppar in den i (1) så får vi $d \leq 2s$, vilket betyder att $\frac{d}{2} \leq s$.
Vi vet också, i och med vårt antagande att $s \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ att det också gäller att

$$s \leq \left\lfloor \frac{d-1}{2} \right\rfloor \leq \frac{d-1}{2} < \frac{d}{2}. \tag{3}$$

Slutligen får vi, om vi kombinerar dessa olikheter, får vi att

$$\frac{d}{2} \leq s < \frac{d}{2}$$

vilket är en motsägelse. Alltså är $\delta(w, c) \neq \delta(w, c^*)$, och koden korrigerar w till den närmaste. □

3 Reed-Solomon-koder

Reed-Solomon-koder är en klass felrättande koder som används inom digital kommunikation och datalagring för att detektera och korrigera fel som uppstår i en datasekvens. Reed-Solomon-koder utvecklades, som namnet antyder, av Irving. S. Reed och Gustave Solomon år 1960. Idag använder vi dessa koder där det är viktigt att datan och meddelandet måste bevaras, till exempel avläsning av CD-skivor, DVD, och som vi kommer gå in på mer i detalj senare QR-koder, men också vid kommunikation med rymdsonderna Voyager 1 och 2 som skickades upp av NASA under 70-talet.
Reed-Solomon-koder bygger på polynominterpolering, där ett meddelande representeras av koefficienter i ett polynom. Detta polynom evalueras sedan på förutbestämda punkter. Poängen är sedan att, med hjälp av dessa punkter, rekonstruera polynomet så att meddelandet kan utläsas. För att lägga till den felrättande aspekten kan fler punkter än symboler i meddelandet läggas till. Då kan polynomet rekonstrueras även om det skulle uppstå fel i dataöverföringen. [Wik24a]

3.1 Lagrange's metod för polynominterpolering

Låt oss anta att vi har $N+1$ punkter $(x_0, y_0), (x_1, y_1), (x_2, y_2), ..., (x_{N-1}, y_{N-1})$, i det kartesiska koordinatsystemet. Låt oss också anta att det för alla x_i gäller att $x_i = x_j$ om och endast om $i = j$. Vårt mål är nu att skapa ett polynom $L(x)$ av grad N sådant att $L(x_k) = y_k$ för alla $0 \leq k \leq N$. För att uppnå vårt mål använder vi Lagrange's metod för polynominterpolering, också kallad Lagrangeinterpolering.

Lagrangeinterpolering kan utföras med hjälp av följande formel: [BKS21]

$$L(x) := \sum_{k=0}^{N} y_k \prod_{\substack{m=0 \\ m \neq k}}^{N} \frac{x - x_m}{x_k - x_m}.$$

Som vid en första anblick kan kännas lite överväldigande. Det blir enklare ifall vi delar upp ekvationen i flera steg. Följande är vad vi har att göra:

- För varje punkt x_k skapar vi ett polynom av grad N sådant att detta polynom har alla andra punkter som nollställen. Eftersom vi har totalt $N+1$ punkter kommer vi alltså få lika många polynom.

- Vi ändrar varje polynom genom att multiplicera detta med en konstant sådant att varje polynom evalueras till 1 för det x_k som inte blir 0. Vi kallar detta för vårt baspolynom $\ell_k(x)$, som alltså uppfyller $\ell_k(x_k) = 1$ och $\ell_k(x_m) = 0$ om $k \neq m$
- Vi skapar sedan $L_k(x) = y_k \cdot \ell_k(x)$. Detta ger oss att $L_k(x_k) = y_k \cdot \ell_k(x_k) = 1 \cdot y_k = y_k$.
- Lagrangepolynomet $L(x)$ är nu summan av alla $L_k(x)$ då $0 \leq k \leq N$.

3.2 Lagrangeinterpolering steg för steg

Låt oss igen anta att vi har $N+1$ punkter $(x_0, y_0), (x_1, y_1), (x_2, y_2), ..., (x_{N-1}, y_{N-1}), (x_N, y_N)$ som uppfyller $x_i \neq x_j$ om $i \neq j$. Vi letar efter ett polynom $L(x)$ av grad N, sådant att $L(x_k) = y_k$ för alla $0 \leq k \leq N$.

Steg 1.
Välj ut en punkt (x_k, y_k).

Steg 2.
Konstruera grundpolynomet
$(x - x_0)(x - x_1)(x - x_2)...(x - x_{k-1})(x - x_{k+1})...(x - x_{N-1})(x - x_N)$.
Detta polynom kommer att evaluera till för alla x_m då $m \neq k$, och kommer att ha grad N.

Steg 3.
Om vi evaluerar ovan polynom över x_k kommer resultatet att bli

$$(x_k - x_0)(x_k - x_1) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_N) \neq 0.$$

Om vi skapar polynomet

$$\ell_k(x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_N)}{(x_k - x_0)(x_k - x_1) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_N)}$$

kan vi alltså vara säkra på att $\ell_k(x_k) = 1$. Detta är ett av våra baspolynom, de byggstenar vi sedan använder för att sedan konstruera Lagrangepolynomet.

Steg 4.
Vi sätter nu $L_k(x) := y_k \cdot \ell_k(x)$. Ur detta kan vi se att

$$L_k(x_k) = y_k \cdot \ell_k(x_k) = 1 \cdot y_k = y_k.$$

Steg 5.

Repetera nu **Steg 1–4** för alla andra punkter.

Steg 6.

Summera nu alla $L_k(x)$ för att till slut få ut $L(x)$. Det gäller alltså att:
$L(x) = \sum_{k=0}^{N} L_k(x)$

Låt oss nu testa denna metod med ett exempel.

Exempel 3.1. Låt oss anta att vi har följande punkter $(1, -4), (2, -1), (3, -6)$ för vilka vi vill hitta ett andragradspolynom.
Vi konstruerar grundpolynomen:
$(x-2)(x-3)$
$(x-1)(x-3)$
$(x-1)(x-2)$.

Vi använder sedan grundpolynomen för att skapa våra baspolynom:
$\ell_1(x) = \frac{(x-2)(x-3)}{(1-2)(1-3)}$
$\ell_2(x) = \frac{(x-1)(x-3)}{(2-1)(2-3)}$
$\ell_3(x) = \frac{(x-1)(x-2)}{(3-1)(3-2)}$.

Vi multiplicerar varje baspolynom med dess motsvarande punkts y-värde:
$L_1(x) = -4 \cdot \frac{(x-2)(x-3)}{(1-2)(1-3)} = -4\left(\frac{1}{2}x^2 - \frac{5}{2}x + 3\right) = -2x^2 + 10x - 12$
$L_2(x) = -1 \cdot \frac{(x-1)(x-3)}{(2-1)(2-3)} = -\left(-x^2 + 4x - 3\right) = x^2 - 4x + 3$
$L_3(x) = 6 \cdot \frac{(x-1)(x-2)}{(3-1)(3-2)} = 6\left(\frac{1}{2}x^2 - \frac{3}{2}x + 1\right) = 3x^2 - 9x + 6.$

Vi summerar nu dessa polynom för att få Lagrangepolynomet:
$L(x) = (-2x^2 + 10x - 12) + (x^2 - 4x + 3) + (3x^2 - 9x + 6) = 2x^2 - 3x - 3$
För att övertyga oss om att Lagrangepolynomet faktiskt träffar våra punkter gör vi en kontrollräkning:
$L(1) = 2 \cdot 1^2 - 3 \cdot 1 - 3 = -4$
$L(2) = 2 \cdot 2^2 - 3 \cdot 2 - 3 = -1$
$L(3) = 2 \cdot 3^2 - 3 \cdot 3 - 3 = 6$
Allt stämmer perfekt!

3.3 Unikheten av Lagrange'spolynom

Reed-Solomon-koder bygger på det faktum att, för ett givet antal $N+1$ punkter så finns det ett och endast ett polynom av grad N som interpolerar dessa punkter. Låt oss bevisa att detta är sant.

Sats 3.2. *Varje Lagrangepolynom är unikt.*

Bevis. Antag att vi har $N+1$ punkter $(x_0, y_0)(x_1, y_1)(x_2, y_2), ..., (x_N, y_N)$ sådant att $x_i = x_j$ endast om $i = j$. Vidare antar vi att det finns två olika Lagrangepolynom $L(x)$ och $L_*(x)$ som båda interpolerar dessa punkter, men sådant att $L(x) \neq L_*(x)$. Värt att poängtera är att båda dessa polynom är av grad N.

Vi kan nu skapa polynomet

$$R(x) = L(x) - L_*(x).$$

Eftersom $R(x)$ är differensen av två polynom av grad N är det värt att notera att grad$R(x) \leq N$. Det kommer för alla x_k där $0 \leq k \leq N$ gälla att

$$L(x_k) = L_*(x_k) = y_k$$

och således kommer

$$R(x_k) = L(x_k) - L_*(x_k) = y_k - y_k = 0.$$

Det gäller alltså att, för alla $N+1$ punkter att gälla att $R(x_k) = 0$. Detta kan, eftersom grad$R(x) \leq N$, bara vara sant ifall $R(x) = 0$ vilket ger oss att $L(x) = L_*(x)$. [?] □

Samma argument håller även för polynom över ändliga kroppar.

3.4 Att koda en Reed-Solomon-kod

Delkapitel 3.5 till 3.8 bygger på ett blogginlägg av Tom Verbeure. [Ver22]
Innan du och en mottagare kan börja skicka Reed-Solomon-koder måste ni komma överens om ett par saker, nämligen

- Längden på ett kodord: n.
- Dimensionen, längden på ett meddelande k.

- Ett alfabet, d.v.s en kropp som bestämmer vilka symboler som finns tillgängliga och de operationer och relationer som finns mellan symbolerna.
- En lista med värden $a_0, a_1, a_2, ..., a_{n-2}, a_{n-1}$ över vilka polynomet ska evalueras.

När ni väl kommit överens om dessa parametrar kodar du ditt meddelande enligt följande:

Steg 1.
Välj ett meddelande $m_0, m_1, ..., m_{k-1}$ sådant att alla m_i ingår i det valda alfabetet.

Steg 2.
Skapa sedan meddelandepolynomet $m_0 + m_1x + m_2x^2 + ... + m_{k-1}x^{k-1}$. Det gäller alltså att $m(x) := \sum_{i=0}^{k-1} m_i x^i$.

Steg 3.
Kodvektorn som du ska skicka till mottagaren är nu $(m(a_0), m(a_1), ..., m(a_{n-1}))$.

3.5 Att avkoda en Reed-Solomon-kod

Låt oss säga att du fått ett meddelande som konstruerats enligt instruktionerna ovan, hur gör vi för att läsa det? Notera att vi har samma överenskomna parametrar som ovan. Reed-Solomon-koder kan detektera och korrigera upp till $\left\lfloor \frac{n-k}{2} \right\rfloor$. Vi antar att det blivit $s \leq \left\lfloor \frac{n-k}{2} \right\rfloor$ fel i överföringen.

Steg 1.
Meddelandevektorn som skickades till oss har n punkter som ser ut på följande vis: $(a_i, m(a_i))$. Ta fram varje urval av k sådana punkter, det kommer att finnas $\binom{n}{k}$ sådana urval. För varje urval, skapa det motsvarande Lagrangepolynomet, som kommer att ha grad $k - 1$.

Steg 2.
Om inga fel har uppstått i överföringen av information så kommer alla punkter att ligga på samma polynom, nämligen $m(x)$, och det är också det resultatet vi kommer att få av Lagrangeinterpoleringen. Därefter kan koefficienterna avläsas och det ursprungliga meddelandet kan återkonstrueras.
Om det däremot någonstans har blivit fel kanske det kan korrigeras. Då kommer inte alla urval ge samma Lagrangepolynom. Det vi gör då är att välja det Lagrangepolynom som skapas av flest urval. Vi låter alltså de här urvalen rösta på vilket polynom de tror är $m(x)$. Detta kommer alltid att fungera om $s \leq \left\lfloor \frac{n-k}{2} \right\rfloor$.

Sats 3.3. *Om vi mottagit en meddelandevektor där* $s \leq \left\lfloor \frac{n-k}{2} \right\rfloor$ *kommer alltid* $m(x)$ *få flest röster.*

Bevis. Detta bevis kommer att gå till som följande:

1. Först kommer vi räkna hur många röster $m(x)$ kommer att få.
2. Sedan kommer vi skapa en annan kandidat, $p(x) \neq m(x)$, och undersöka om det finns en övre gräns för antalet röster $p(x)$ kan få.
3. Sedan visar vi att $m(x)$ kommer att få fler röster än det övre taket för antalet röster för $p(x)$.

1. Vi börjar med att titta på hur många röster $m(x)$ kommer att få. Om det uppstått s fel i överföringen betyder det att vi har $n-s$ korrekta punkter kvar. Vilket urval av k stycken av de korrekta punkterna kommer att resultera i $m(x)$. Ett sådant urval kan ske på $\binom{n-s}{k}$ sätt, och således är det också så många röster $m(x)$ kommer att få.

2. För att $p(x)$ ska få så många röster som möjligt måste så många punkter som möjligt ligga på $p(x)$, alltså att de uppfyller att $p(x_i) = y_i$. Detta sker om $k-1$ korrekta punkter ligger på $p(x)$ [2] och om alla inkorrekta punkter också ligger på $p(x)$, alltså totalt $s+k-1$ punkter. Polynomet $p(x)$ kommer således att få totalt $\binom{s+k-1}{k}$ röster.

3. Sista steget är nu att visa att $\binom{n-s}{k} > \binom{s+k-1}{k}$. Eftersom både $n-s$ och $s+k-1$ står över k så räcker det med att visa att

$$n - s > s + k - 1$$

som vi kan skriva om som

$$n - k > 2s - 1. \tag{1}$$

Slutligen vet vi att

$$s \leq \left\lfloor \frac{n-k}{2} \right\rfloor \leq \frac{n-k}{2} \implies 2s \leq n - k. \tag{2}$$

[2] Kom ihåg att Lagrangepolynom är unika, så om alla korrekta punkter hade legat på $p(x)$ hade det gällt att $p(x) = m(x)$

Om vi stoppar in (2) i (1) så får vi olikheten

$$n - k \geq 2s > 2s - 1$$

och vi är därmed klara. □

3.6 Exempel på kodning och avkodning av Reed-Solomon-koder

Låt oss ta ett exempel på kodning och avkodning av Reed-Solomon-koder. Vi bestämmer först $n = 4$, $k = 2$, att vi använder de rationella talen, och att vi ska evaluera polynomet över 0,1,2,3.

Vi börjar med att välja meddelandet 6,9, som vi kodar som polynomet $m(x) = 6+9x$. Då skickar vi sedan koden $(m(0), m(1), m(2), m(3)) = (6, 15, 24, 33)$. Vår kod kan rätta upp till 1 fel.

Vår mottagare tar emot $(6, 15, 18, 33)$. Vi vet att något blivit fel, nämligen att 24 bytts ut mot 18, alltså att det uppstått ett fel. Det vet förstås inte mottagaren, än iallfall. Inga problem. Antalet fel är inom felrättningskapaciteten för vår kod. Låt oss ta fram alla möjliga urval av punkter, sätta in dem i Lagrange's formel, och så får de rösta om vad meddelandet ska vara.

Punkter	Lagrangepolynom
(0,6), (1,15)	$6 + 9x$
(0,6), (2,18)	$6 + 6x$
(0,6), (3,33)	$6 + 9x$
(1,15), (2,18)	$12 + 3x$
(1,15), (3,33)	$6 + 9x$
(2,18), (3,33)	$-12 + 15x$

Vi ser att $6 + 9x$ fick flest röster, och det var också det meddelande vi skickade, så vår mottagare fick det korrekta meddelandet till slut!

3.7 Systematiska Reed-Solomon-koder

Metoden för kodning och avkodning av Reed-Solomon-koder vi precis gick igenom är relativt enkel, och i teorin fungerar den bra. Det finns dock utrymme för förbättring.

Det stora problemet ligger i hur många urval av punkter som kan uppstå ifall n och k, är väldigt stora. Då blir röstningsprincipen opraktisk, eftersom den helt enkelt tar för lång tid för att kunna genomföra i verkligheten. Detta kan vi lösa genom att introducera systematiska koder.

Definition 3.4. En kod kallas systematisk om det ursprungliga meddelandet finns med som en del av kodordet. [Wik23]

I fallet Reed-Solomon-koder betyder det att kodordet kommer bli på formen $m_0, m_1, ...m_{k-1}, r_k, ..., r_{n-1}$, där m_i är en del av meddelandet, och r_i är redundans. Det enklaste sättet att bearbeta den icke-systematiska metoden ovan hade varit att, istället för att låta

$$m(x) := \sum_{i=0}^{k-1} m_i x^i$$

, och sedan skicka $m(a_0), m(a_1), ..., m(a_{n-1})$, så kan vi istället låta $m(x)$ vara det Lagrange'spolynom som interpolerar punkterna (a_i, m_i) för $0 \leq i \leq k-1$. Vi sätter sedan $r_i = m(a_i)$ när $k \leq i \leq n-1$, och skickar precis $m_0, m_1, ...m_{k-1}, r_k, ..., r_{n-1}$.

3.8 Kodning av systematiska BCH-Reed-Solomon-koder

Den systematiska metoden ovan är en marginell förbättring, men löser inte problemet med stora n och k, så låt oss diskutera en metod som faktiskt gör det, nämligen BCH–Reed-Solomon-koder. BCH-koder[3] innebär att vi använder oss av en cyklisk ändlig kropp och ett generatorpolynom
Innan vi börjar med våra förbättrade, systematiska Reed-Solomon-koder behöver vi, precis som förr komma överens om några saker innan vi börjar. Den listan ser dock lite annorlunda ut för systematiska koder.

- Längden på ett kodord: n
- Dimensionen: k
- Ett alfabet som är en ändlig kropp $\mathbb{F}$. Detta är vad som gör koden till en BCH-kod.
- En primitiv rot α till $\mathbb{F}$.

[3]en förkortning av Bose–Chaudhuri–Hocquenghem-koder, döpt efter Alexis Hocquenghem, Raj Chandra Bose, och wijendra Kumar Ray-Chaudhuri

- En lista med värden $a^0, a^1, a^2, ..., a^{n-k-1}$ som generar redundansen

Planen är att skapa ett polynom $m(x)$ som har meddelandet som koefficienter, och har egenskapen att $m(a_i) = 0$. Låt oss börja!

Steg 1.
Välj ett meddelande $m_0, m_1, ..., m_{k-1}$ sådant att alla m_i ingår i det valda alfabetet.

Steg 2.
Skapa polynomet $p(x) = m_0 + m_1x + .. + m_{k-1}x^{k-1} = \sum_{i=0}^{k-1} m_i x^i$.

Steg 3.
Skapa polynomet $g(x) = (x-1)(x-a)(x-a^2)\cdots(x-a^{n-k-1})$. $g(x)$ kommer att ha grad $n-k$, och är konstruerat på så vis att $g(a^i) = 0$.

Steg 4.
Vi konstruerar nu polynomet

$$p(x)x^{n-k} = m_0x^{n-k} + m_1x^{n-k+1} + m_2x^{n-k+2} + ... + m_{k-1}x^{n-1}.$$

Steg 5.
Nu utför vi polynomdivision $\frac{p(x)x^{n-k}}{g(x)}$ på sådant vis att $p\,(x)\,x^{n-k} = q(x)g(x) + r(x)$ och $\deg(r) < \deg(g)$.

Steg 6.
Ansätt nu $m(x) := p\,(x)\,x^{n-k} - r(x) = q(x)g(x)$. Eftersom $g(x)$ är en faktor är det enkelt att se att $m(a^i) = 0$.
Eftersom $g(x)$ har konstaterat grad $n-k$, och $r(x)$ är konstruerad på så vis att $\deg(r) < \deg(g)$ så kommer alltså

$$m(x) = r_0 + r_1x + ... + r_{n-k-1}x^{n-k-1} + m_0x^{n-k} + ... + m_{k-1}x^{n-1}.$$

Vårt kodord blir nu $m_0, m_1, ..., m_{k-1}, r_k, ..., r_{n-1}$.

3.9 Avkodning av systematiska BCH-Reed-Solomon-koder

[Wik24a] Vi kommer nu utföra en avkodning av koden skapad ovan med hjälp av Peterson-Gorenstein-Zierler-avkodning(PGZ-avkodning). Metoden går ut på att göra följande.

- Antag att vi tagit emot kodord $m^*(x) = m(x) + e(x)$ där $e(x)$ är felpolynomet av grad $n-1$.
- Beräkna $S_j = m^*(\alpha^j) = e(\alpha^j)$ för alla $1 \leq j \leq n-k$
- Hitta var felen uppstått med hjälp av ett felfinnarpolynom $\Lambda(x)$.
- Hitta värdena på de fel som uppstått.
- Rekonstruera $m(x)$ med kunskapen om var felen uppstått, och hur stora de är.

Steg 1.
Antag att vi tagit emot ett kodord $m_0^*, m_1^*, ..., m_{k-1}^*, r_k^*, ..., r_{n-1}^*$. Kasta om termerna så att de hamnar i rätt ordning, och vi får det mottagna meddelandepolynomet

$$m^*(x) = r_0^* + r_1^* x + ... + r_{n-k-1}^* x^{n-k-1} + m_0^* x^{n-k} + ... + m_{k-1}^* x^{n-1}.$$

Steg 2.
Vi skapar nu felpolynomet $e(x)$ av grad $n-1$ sådant att $m^*(x) = m(x) + e(x)$. Om $e_i \neq 0$ innebär det att det uppstått ett fel på position i, men än så länge vet vi inte $e(x)$.
Steg 3. Nu beräknar vi alla syndrom som tillhör $m^*(x)$. Vi definierar syndromet S_j enligt följande: $S_j = m^*(\alpha^j) = m(\alpha^j) + e(\alpha^j)$ för $1 \leq j \leq n-k$, men eftersom[4] $m(\alpha^j) = 0$ så får vi $S_j = e(\alpha^j)$
Steg 4. Vi kommer nu att anta att det uppstått s stycken fel i överföringen. Vidare antar vi att dessa fel uppstått på index $i_1, i_2, ..., i_k, ..., i_s$ där $1 \leq k \leq s$. Vi behöver nu definiera två ytterligare koncept. Först ut är $X_k := \alpha^{i_k}$ Dessa X_k är våra felfinnare. Om vi lyckas lista ut värdet på X_k så vet vi att $i_k = \log_\alpha(X_k)$, och vi vet därmed positionen av det k:e felet.
Sedan definierar vi $Y_k = e_{i_k}$ alltså storleken på det k:e felet. Vi vet att
$m_{i_k}^* = m_{i_k} + e_{i_k} = m_{i_k} + Y_k$. Ifall vi kan lista ut både X_k och Y_k kan vi korrigera det k:e felet.

[4]Kom ihåg att vi definierade $g(x) = (x - a^1)(x - a^2)...(x - a^{n-k})$, så $g(\alpha^j) = 0$ och således är även $m(\alpha^j) = q(\alpha^j)g(\alpha^j) = 0$.

Steg 5.
Vi kan nu konstatera att

$$S_j = e\left(\alpha^j\right)\sum_{k=1}^{s} e_{i_k}\left(\alpha^j\right)^{i_k} = \sum_{k=1}^{s} Y_k X_k^j \tag{1}$$

Vilket, om vi observerar att varje j ger oss en ekvation, gör att vi kan sätta upp ekvationssystemet:

$$\begin{aligned} X_1^1 Y_1 + X_2^1 Y_2 + ... + X_s^1 Y_s &= S_1 \\ X_1^2 Y_1 + X_2^2 Y_2 + ... + X_s^2 Y_s &= S_2 \\ &\vdots \\ X_1^{n-k} Y_1 + X_2^{n-k} Y_2 + ... + X_s^{n-k} Y_s &= S_{n-k}. \end{aligned}$$

För att kunna lösa ekvationssystemet måste vi först bestämma alla X_k[5].

Steg 6.
För att underlätta att hitta X_k inför vi felfinnarpolynomet

$$\Lambda(x) = (1 - xX_1)(1 - xX_2)...(1 - xX_s) = 1 + \Lambda_1 x + \Lambda_2 x^2 + ... + \Lambda_s x^s$$

med rötter $\Lambda(X_k^{-1}) = 0$. Om vi kan bestämma koefficienterna till $\Lambda(x)$ kan vi hitta dess rötter, och då lista ut alla X_k. Koefficienterna till $\Lambda(x)$ är relaterade till våra syndrom enligt

$$S_j\Lambda_s + S_{j+1}\Lambda_{s-1} + ... + S_{j+v-1}\Lambda_1 = -S_{j+s} \tag{2}$$

vilket, då vi återigen itererar över $1 \leq j \leq n - k$ ekvationssystemet

$$\begin{aligned} S_1\Lambda_s + S_2\Lambda_{s-1} + ... + S_s\Lambda_1 &= -S_{s+1} \\ S_2\Lambda_s + S_3\Lambda_{s-1} + ... + S_{s+1}\Lambda_1 &= -S_{s+2} \\ &\vdots \\ S_s\Lambda_s + S_{s+1}\Lambda_{s-1} + ... + S_{2s-1}\Lambda_1 &= -S_{2s}. \end{aligned}$$

Detta ekvationssystem antar att s är känt, vilket det inte nödvändigtvis är. Då

[5]Vi har faktiskt $n - k$ ekvationer och $2s$ okända, alltså går ekvationssystemet att lösa om $n - k \geq 2s$, men eftersom det är icke-linjärt vore det lättare att först ta reda på alla X_k, och sedan ge sig på att lösa (1).

börjar vi med att anta att $2s = n - k$, och om det ekvationssystemet ej har en lösning minskar prövar vi med att minska s med 1, och gör det succesivt till dess att vi hittat lösningen.

Steg 7.
När vi har funnit $\Lambda_1, \Lambda_2, ..., \Lambda_s$ hittar vi rötterna till $\Lambda(x)$ och får på så sätt ut alla X_k.

Steg 8.
Använd alla X_k för att lösa (1), och få ut alla Y_k.

Steg 9.
Beräkna alla $i_k = \log_\alpha(X_k)$.

Steg 10.
Vi kan nu beräkna

$$e(x) = \sum_{k=1}^{s} e_{i_k} x^{i_k} = \sum_{k=1}^{s} Y_k x^{i_k}$$

Vi kan nu få fram $m(x)$ genom att ta $m^*(x) - e(x)$.

Härledning av (2): [Wik24a]
Vi hade vår utgångspunkt

$$\begin{aligned}\Lambda(X_k^{-1}) = \\ \left(1 - X_k^{-1}X_k\right)\left(1 - X_k^{-1}X_1\right)...\left(1 - X_k^{-1}X_{k-1}\right)\left(1 - X_k^{-1}X_{k+1}\right)...\left(1 - X_k^{-1}X_s\right) = \\ (1-1)\left(1 - X_k^{-1}X_1\right)...\left(1 - X_k^{-1}X_{k-1}\right)\left(1 - X_k^{-1}X_{k+1}\right)...\left(1 - X_k^{-1}X_s\right) = 0.\end{aligned}$$

En multiplikation med $Y_k X_k^{j+s}$ på båda sidor ger oss.

$$Y_k X_k^{j+s} \Lambda(X_k^{-1}) = 0$$

Om vi utvecklar $\Lambda(X_k^{-1}) = 1 + \Lambda_1 X_k^{-1} + \Lambda_2 X_k^{-2} + ... + \Lambda_s X_k^{-s}$ och distribuerar termerna så får vi

$$\begin{aligned}Y_k X_k^{j+s}(1 + \Lambda_1 X_k^{-1} + \Lambda_2 X_k^{-2} + ... + \Lambda_s X_k^{-s}) = \\ Y_k X_k^{j+s} + Y_k X_k^{j+s}\Lambda_1 X_k^{-1} + Y_k X_k^{j+s}\Lambda_2 X_k^{-2} + ... + Y_k X_k^{j+s}\Lambda_s X_k^{-s}) = \\ Y_k X_k^{j+s} + \Lambda_1 Y_k X_k^{j+s-1} + \Lambda_2 Y_k X_k^{j+s-2} + ... + \Lambda_s Y_k X_k^{j} = 0\end{aligned}$$

Vidare kan vi konstatera att eftersom $\Lambda(X_k^{-1}) = 0$ för alla $1 \leq k \leq s$ så gäller det även att $\sum_{k=1}^{s} \Lambda\left(X_k^{-1}\right) = 0$, och det ger oss.

$$\sum_{k=1}^{s}(Y_k X_k^{j+s} + \Lambda_1 Y_k X_k^{j+s-1} + \Lambda_2 Y_k X_k^{j+s-2} + ... + \Lambda_s Y_k X_k^{j}) = 0.$$

Nu buntar vi ihop varje term till sin egen summa, och bryter ut Λ-koefficienten som en konstant.

$$\sum_{k=1}^{s}(Y_k X_k^{j+s} + \Lambda_1 Y_k X_k^{j+s-1} + \Lambda_2 Y_k X_k^{j+s-2} + ... + \Lambda_s Y_k X_k^{j}) =$$
$$\sum_{k=1}^{s}(Y_k X_k^{j+s}) + \sum_{k=1}^{s}(\Lambda_1 Y_k X_k^{j+s-1}) + \sum_{k=1}^{s}(\Lambda_2 Y_k X_k^{j+s-2}) + ... + \sum_{k=1}^{s}(\Lambda_s Y_k X_k^{j}) =$$
$$\sum_{k=1}^{s}(Y_k X_k^{j+s}) + \Lambda_1 \sum_{k=1}^{s}(Y_k X_k^{j+s-1}) + \Lambda_2 \sum_{k=1}^{s}(Y_k X_k^{j+s-2}) + ... + \Lambda_s \sum_{k=1}^{s}(Y_k X_k^{j}) = 0.$$

Slutligen påminner vi oss själva om att vi definierade $S_j = \sum_{k=1}^{s} Y_k X_k^j$. En omskrivning senare får vi

$$\sum_{k=1}^{s}(Y_k X_k^{j+s}) + \Lambda_1 \sum_{k=1}^{s}(Y_k X_k^{j+s-1}) + \Lambda_2 \sum_{k=1}^{s}(Y_k X_k^{j+s-2}) + ... + \Lambda_s \sum_{k=1}^{s}(Y_k X_k^{j}) =$$
$$S_{j+s} + S_j\Lambda_s + S_{j+1}\Lambda_{s-1} + ... + S_{j+v-1}\Lambda_1 = 0$$

Vi subtraherar S_{j+s} från båda sidor vilken slutligen tar oss i mål.

$$S_j\Lambda_s + S_{j+1}\Lambda_{s-1} + ... + S_{j+v-1}\Lambda_1 = -S_{j+s}. \tag{2}$$

3.10 Exempel på systematiska BCH-Reed-Solomon-koder

Vi prövar att använda den här metoden också. Vi väljer, precis som förr, $n = 4$, $k = 2$, vi väljer vår ändliga kropp till heltalen modulo 5, och vår primitiva rot $\alpha = 2$.

Vi väljer meddelandet $3, 4$, vilket ger oss att $p(x) = 3 + 4x$. Vi kan samtidigt skapa generatorpolynomet

$$g(x) = (x - 1)(x - 2^1) = x^2 - 3x + 2 \equiv_5 x^2 + 2x + 2.$$

Sedan är det dags att utföra polynomdivisionen $\frac{p(x)x^{n-k}}{g(x)}$ som ger oss resten $0 + 2x$.

Alltså får vi ett meddelandepolynom

$$m(x) = -0 - 2x + 3x^2 + 4x^3 = 0 + 3x + 3x^2 + 4x^3.$$

Meddelandepolynomet $m(x) = 0+3x+3x^2+4x^3$, så vi skickar det till vår mottagare.

Låt oss säga att vår mottagare tar emot polynomet $m^*(x) = 1 + 3x + 3x^2 + 4x^3$. Det har återigen uppstått ett fel. Dags att avkoda! Vi börjar med att anta att $m^*(x) = m(x) + e(x)$. Sedan är det dags att beräkna syndromen $S_1 = m^*(2^0) = 1$ och $S_2 = m^*(2^1) = 1$. Eftersom båda är nollskilda kan mottagaren anta att något blivit fel och gå vidare med avkodningen.

Nu inför vi felfinnarpolynomet, för att få reda på var felet har uppstått. Vi vet, enligt konstruktionen av vår kod, att vi bara kan korrigera upp till ett fel, och vi vet att det har blivit ett fel eftersom syndromen är nollskilda. Felfinnarpolynomet kommer därmed att bli $\Lambda(x) = 1 - xX_1$. Eftersom vi bara har ett enda fel kommer vårt ekvationssystem bara bestå av en ekvation, nämligen $S_1\Lambda_1 = -S_2$. Sätter vi in $S_1 = S_2 = 1$ så får vi $\Lambda_1 = -1 \equiv_5 4$. Alltså gäller det att $\Lambda(x) = 1 + 4x$. Om vi löser ekvationen $\Lambda(x) = 1 + 4x = 0$ får vi att $x = 1$. Vi vet också att roten till $\Lambda(x)$ är X_1^{-1}, och $1^{-1} = 1$.. Kom ihåg att vi definierade $X_k = \alpha^{i_k}$, som ger oss att $X_1 = \alpha^{i_1} = 1$, s¨$i_1 = 0$. Vi kan då dra slutsatsen att det har hänt ett fel i termen av grad 0.

Nu är det dags att undersöka vad felet är. Vi sätter upp ekvationen $X_1^1 Y_1 = S_1$. Insätting av X_1 och S_1 ger $1Y_1 = 1$ som ger att även $Y_1 = 1$. Då vet vi även vad felet är, och vi kan korrigera det genom att ta $m^*(x) - 1 = 0 + 3x + 3x^2 + 4x^3$, ur vilket vi kan läsa meddelandet $3, 4$, och då är vi klara.

3.11 Primitiva polynom för att generera $\mathbb{F}_{p^q}$

I kodteorin skapar man ofta ändliga kroppar med hjälp av primitiva polynom, som enkelt låter oss skapa kroppen $\mathbb{F}_{p^q}$, där p är ett primtal, och $q > 1$ är ett heltal. Till skillnad från att enbart räkna med heltal modulo n, kan vi lättare välja en storlek på en kropp som passar just vårt användningsområde. Kom ihåg att heltalen modulo n enbart är en grupp ifall n är ett primtal. Använder vi istället primitiva polynom så kan storleken på vår kropp även vara potenser av primtal, d.v.s. p^q. Ofta används just $p = 2$ eftersom det låter oss använda det binära talsystemet, något som

underlättar ifall den ändliga kroppen ska användas av digitala maskiner.

Definition 3.5. Ett primitivt polynom är ett moniskt och irreducibelt polynom med koefficienter i $\mathbb{F}_p$ och grad q som kan användas för att skapa den ändliga gruppen $\mathbb{F}_{p^q}$. [Wik24b]

Planen är att ta ett primitivt polynom $p(x) = a_0 + a_1x + a_2x^2 + \cdots + x^q$ där a_i med $0 \leq i \leq q-1$, anta att det finns en rot $\alpha \in \mathbb{F}_{p^q}$, som också är en primitiv rot till $\mathbb{F}_{p^q}$, och använda lösningen $p(\alpha) = 0$ för att generera vår önskade ändliga kropp med element $\left\{0, 1, \alpha, \alpha^2, \alpha^3, ..., \alpha^{p^q-2}\right\}$. Låt oss ta ett exempel.

Exempel 3.6. Vi använder oss av $p(x) = x^3 + x + 1$ över $\mathbb{F}_2$ för att generera $\mathbb{F}_{2^3}$, eftersom 3 är graden på vårt primitiva polynom $p(x)$.

Vi börjar med att ansätta att $p(\alpha) = 0$. Det ger oss

$$\alpha^3 + \alpha + 1 = 0 \implies$$
$$\alpha^3 = -\alpha - 1 = \alpha + 1.$$

Notera att den sista omskrivningen $-\alpha - 1 = \alpha + 1$ är något vi kan göra eftersom våra koefficienter är hämtade från kroppen heltalen modulo 2, där addition och subtraktion ger samma resultat.

Dags att skapa vår grupp.

Element	Uttryckt i alpha
0	0
1	1
α	α
α^2	α^2
α^3	$\alpha^3 = 1 + \alpha$
α^4	$\alpha^4 = \alpha(1 + \alpha) = \alpha + \alpha^2$
α^5	$\alpha^5 = \alpha\left(\alpha + \alpha^2\right) = \alpha^2 + \alpha^3 = \alpha^2 + \alpha + 1$
α^6	$\alpha^6 = \alpha\left(\alpha^2 + \alpha + 1\right) = \alpha^3 + \alpha^2 + \alpha = 1 + \alpha + \alpha^2 + \alpha \ = \ 1 + \alpha^2$

och där var vi i mål. Nu har vi en cyklisk, ändlig kropp av storlek 2^3. Addition under gruppen fungerar precis som vanlig polynomaddition modulo 2, d.v.s. kommer till

exempel

$$\alpha^3 + \alpha^5 = (\alpha + 1) + (\alpha^2 + \alpha + 1) = \alpha^2 + 2\alpha + 2 = \alpha^2.$$

Enklare vore ju förstås att representera varje potensuttryck av α som en vektor i $\mathbb{F}_2^3$ där $\beta_2\alpha^2 + \beta_1\alpha + \beta_0$ skulle skrivas som $(\beta_2, \beta_1, \beta_0)$, där varje $\beta_i \in \mathbb{F}_2$. Då får man ett väldigt naturligt sätt att representera varje element i den skapade cykliska ändliga kroppen binärt, som kan vara bra om man till exempel jobbar med ASCII. Detta är vad som i en spännande fantasyroman brukar kallas för *foreshadowing*.

4 QR-koder

QR-koder, en förkortning för Quick Response, är en tvådimensionell utveckling av den endimensionella streckkoden. QR-koder är gjorda för att snabbt och enkelt kunna skannas med hjälp av en smartphone, surfplatta, eller en speciell QR-läsarmaskin. Ursprungligen utvecklades QR-koder av det japanska företaget Denso Wave för att enkelt hålla reda på bildelar under tillverkningsprocessen. [Mul24]
Under de trettio åren sedan dess har QR-koder spridit sig utanför bildindustrin, och används idag för att sprida internetlänkar eller ge information åt mobiltelefoner. Det kan till exempel vara en restaurang som har en QR-kod som leder till deras meny, eller någon som satt upp en QR-kod hemma för att underlätta för gäster när de ska ansluta sig till WiFi-nätverket. I detta kapitel kommer vi att titta på hur en QR-kod genereras, och vi kommer också steg för steg konstruera en egen QR-kod, vilket till stor del kommer att bygga på bloggen Project Nayuki, mer specifikt inlägget "Creating a QR Code step by step."[Nay]
Att skapa en QR-kod för hand kanske låter jättesvårt, men om vi bryter ned processen till ett fåtal steg, så är det faktiskt inte så komplicerat. Om man inte är så insatt i QR-koder kan det såklart också låta väldigt lätt att konstruera en QR-kod, lite som att säga "hur svårt kan det egentligen vara att bygga en månraket,men vi kommer att se att det krävs lite mer av oss än att bara sätta ihop förutbestämda pusselbitar. Under QR-kodsbyggandet kommer vi att dyka djupt i hur varje steg går till, från att välja ett meddelande, till att lägga till de felrättande bitarna, till att slutligen fullborda koden så att den går att skanna. Följande är vad vi kommer att göra:

1. Konvertera meddelandet till data och anpassa det till rätt QR-kodversion.
2. Slå samman meddelandedatan med andra förbestämda datasegment och dela upp det i block, om det så skulle behövas.
3. Lägg till felkorrigering i datan.
4. Rita de fasta mönstren.
5. Rita datan (meddelande + förbestämd data + felkorrigering).
6. Hitta det bästa möjliga maskeringsmönstret.
7. Lägg till och rita formatinformationen.

Och när allt det är klart har du en fullt fungerande QR-kod!

4.1 Att välja meddelande och QR-kodversion

När vi skapar en QR-kod är såklart det viktigaste steget att välja vad QR-koden ska säga, det meddelande som ska framföras till den som skannar koden. I detta exempel, alltså under vår hand för hand-konstruktion av en QR-kod kommer vi att använda meddelandet *Alvin* — mitt namn. Det är ett val som på ytan ser enkelt och tråkigt ut, men det finns några anledningar till att använda ett så kort ord, det är nämligen alldeles för kort för ens de minsta QR-koderna.
Storleken på en QR-kod bestäms av versionen på QR-koden. En QR-kod är alltid kvadratisk, förutom när den inte är det. Ett nytt påfund från 2022 av samma företag som kallas Rectangular Micro QR Code kan, som man hör på namnet, vara rektangulär.
En standard kvadratisk QR-kod är som minst 21 pixlar hög och 21 pixlar bred. En sådan QR-kod kallas för version 1. Skulle man utöka båda sidorna med 4 pixlar vardera skulle man få en 25×25-kod, och den kallas för version 2. Den största möjliga QR-koden är version 40 som är 177×177 pixlar stor. Skulle man vara matematiskt lagd kan man säga att Antal pixlar = $(21 + 4\times\text{Versionsnumer})^2$.
Men hur mycket data kan en QR av en viss storlek? Svaret är att det beror på något som kallas för felkorrigeringsnivå. När man skapar sin QR-kod får man välja ungefär hur stor andel av koden som ska ägnas åt felkorrigering. Det finns totalt fyra val, låg(L), medium(M), kvartil(Q), hög(H). Tabellen nedan visar ungefär hur många bytes som kan avläsas fel, och ändå ge oss rätt meddelande, men viktigt att tänka på är att värdena är approximativa. Här måste man ju såklart göra en avvägning. Väljer vi en högre felkorrigeringsnivå så måste vi använda en större kod för att få plats med all information, men vi kan också vara säkrare på att den skannas korrekt. Vi kommer att använda nivå M i det här exemplet, för lagom är alltid bäst.

Felkorrigeringsnivå Låg(L) kan korrigera upp till 7% av bytes.
Felkorrigeringsnivå Medium(M) kan korrigera upp till 15% av bytes.
Felkorrigeringsnivå Kvartil(Q) kan korrigera upp till 25% av bytes.
Felkorrigeringsnivå Hög(H) kan korrigera upp till 30% av bytes.

En QR-kod version 1 med felkorrigeringsnivå M har kapacitet för 14 symboler, alltså mer än tillräckligt för att kunna skriva *Alvin*.

Symbol	Decimal	Binär
A	65	01000001
l	108	01101100
v	118	01110110
i	105	01101001
n	110	01101110

Nu är det dags att konvertera meddelandet till binärt. Detta görs med en ASCII-tabell. Egentligen används ISO/IEC 8859-1, men för det engelska alfabetet är de ekvivalenta.

Vårt meddelande är nu $(65, 108, 118, 105, 110)_{10}$ eller
$(01000001, 01101100, 01110110, 01101001, 01101110)_2$

4.2 Byte padding, och konstruktion av datasträng

Eftersom vårt meddelande *Alvin* är för kort för en QR-kod version 1 med felkorrigeringsnivå M så behöver vi lägga till byte padding, alltså lägga till fler bytes så att vi fyller ut hela kapaciteten, alltså 14 bytes. Detta gör vi genom att, på slutet, lägga till 11101100 och 00010001 alternerande tills dess att vi uppnår totalt 14 bytes.

Innan vi kan konstruera hela datasträngen, och skicka den vidare till felkorrigeringskapitlet, så måste vi lägga till lite kontext i början av meddelandet, nämligen vilket alfabet vi använt och hur många symboler mottagaren ska förvänta sig.
För det förstnämnda så är det förutbestämt att byte mode representeras av 0100. Antalet symboler i meddelandet, utan byte padding, representeras av en byte, och är bara antalet symboler representerat binärt. $5_{10} = 00000101_2$.

Sedan måste vi till slut lägga till en terminator eller avslutare. Den är också standardiserad, och är 0000.

Den här gången använde vi alfabetet byte mode, vilket innebär att alla symboler kommer att kunna representeras med exakt 8 bitar. Så måste det inte nödvändigtvis vara. Till exempel kan man skapa QR-koder med bara siffror, och på så sätt spara lite plats. Då kan 3 siffror få plats på 10 bitar. Använder man istället japanska kanji behövs 13 bitar per symbol. Då måste inte nödvändigtvis meddelandets längd, mätt i bitar, vara en multipel av 8. Alltså inte ett helt antal bytes. Då måste bit padding läggas till. Det görs genom att fylla på med 0:or tills antalet byte är ett heltal.

Segment	Datasträng	Längd(bit)	Längd(byte)
Alfabet	0100	4	1/2
Symbolantal	00000101	8	1
Meddelande	01000001 01101100 01110110 01101001 01101110	40	5
Terminator	0000	4	1/2
Bit padding		0	0
Byte padding	11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100	72	9

Vi har nu alla nödvändiga delar för att konstruera vår datasträng. Detta görs genom att sammanfoga de olika segmenten i följande ordning.

Om vi fogar alla segmenten samman får vi
$(01000000, 01010100, 00010110, 11000111, 01100110, 10010110, 11100000, 11101100, 00010001, 11101100, 00010001, 11101100, 00010001, 11101100, 00010001, 11101100)_2$.
Vi konverterar tillbaka till decimalt för att underlätta i nästa steg då vi ska lägga till felkorrigeringen.
$(64, 84, 22, 199, 102, 150, 224, 236, 17, 236, 17, 236, 17, 236, 17, 236)_{10}$
Lägg märke till hur vårt originella meddelande inte syns i sekvensen! Detta gör QR-koder mycket svåra att läsa av för människor, även om de kan alla knepen bakom hur en QR-kod skapas.

4.3 Felkorrigering

Detta delkapitel bygger på Thonky's QR Code tutorial.[Tho22]
Nu är det dags att göra vår QR-kod lite mer robust, med hjälp av felkorrigering. Vi använder såklart Reed-Solomon-koder, som vi tillägnat ett helt kapitel. Vi kommer också att använda BCH-kodning, alltså den sista metoden som vi går igenom i kapitel 3. En QR-kod version 1 med felkorrigeringsnivå M kräver 10 felkorrigeringssymboler. Vi går igenom exakt samma steg som vi gjorde i delkapitel, men innan vi kan börja måste vi ju först påminna oss om några parametrar för koden.

Dimension på kodordet $k = 16$ symboler.
Längden på kodordet $n = 16 + 10 = 26$.
Ett alfabet/en ändlig kropp. Standard för QR-koder är att välja den kropp som genereras med hjälp av det primitiva polynomet $x^8 + x^4 + x^3 + x^2 + 1 = 0$ över $\mathbb{F}_2$,

för att skapa $\mathbb{F}_{256}$.
Ett primitivt polynom i vår kropp. När vi skapar den antar vi att den innehåller ett tal α som genererar kroppen genom relationen $\alpha^8 = \alpha^4 + \alpha^3 + \alpha^2 + 1$. Det gör det naturligt att välja α som standard primitiv rot.
En lista med tal i vår kropp för att generera redundansen. Dessa är också standardiserade till att vara $1, \alpha, \alpha^2, ..., \alpha^{n-k-1}$ för enkelhetens skull.

Låt oss börja koda!

Steg 1.
Vi börjar som vi kanske kommer ihåg, med att konstruera ett polynom $p(x)$ utifrån vårt meddelande. Notera att detta inte är vårt meddelandepolynom, utan det är det vi ska komma fram till. Vårt meddelande är:
$(64, 84, 22, 199, 102, 150, 224, 236, 17, 236, 17, 236, 17, 236, 17, 236)_{10}$.

Steg 2.
Om vårt meddelande ser ut som ovan kommer vårt meddelandepolynom
$p(x) = \sum_{i=0}^{k-1} m_i x^i$, och i vårt exempel kommer det att se ut som följande:

$$64x^{15} + 84x^{14} + 22x^{13} + 199x^{12} + 102x^{11} + 150x^{10} + 224x^9 + 236x^8 + 17x^7 + \\ +236x^6 + 17x^5 + 236x^4 + 17x^3 + 236x^2 + 17x + 236.$$

Steg 3.
Det är nu dags att skapa generatorpolynomet $g(x) = (x-1)(x-\alpha)\cdots(x-\alpha^{n-k-1})$.
I vårt fall är $n = 26$ och $k = 16$. Alltså får vi att $n - k - 1 = 9$, vilket ger oss generatorpolynomet

$$g(x) = (x-1)(x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)(x-\alpha^5)(x-\alpha^6)(x-\alpha^7)(x-\alpha^8)(x-\alpha^9).$$

Om vi sedan utvecklar generatorpolynomet får vi

$$x^{10} + \alpha^{251}x^9 + \alpha^{67}x^8 + \alpha^{46}x^7 + \alpha^{61}x^6 + \alpha^{118}x^5 + \alpha^{70}x^4 + \alpha^{64}x^3 + \alpha^{94}x^2 + \alpha^{32}x + \alpha^{45}.$$

Steg 4.
Vi förbereder oss nu på att genomföra polynomdivisionen genom att bilda polynomet

$$p(x)x^{n-k} = p(x)x^{10} = 64x^{25} + 84x^{24} + 22x^{23} + \cdots + 17x^{11} + 236x^{10}.$$

Steg 5.

Nu är det dags för den längsta delen, nämligen polynomdivisionen[6]. Vi ska räkna ut $\frac{p(x)x^{10}}{g(x)}$. Hittills har vi skrivit $p(x)$ på decimalform, låt oss använda den här tabellen i appendix för att konvertera koefficienterna till potenser av α. Då får vi:

$$p(x)x^{10} = \alpha^{6}x^{25} + \alpha^{143}x^{24} + \alpha^{239}x^{23} + \alpha^{118}x^{22} + \alpha^{126}x^{21} + \alpha^{180}x^{20} + \alpha^{203}x^{19} + \alpha^{122}x^{18} + \alpha^{100}x^{17} + \alpha^{122}x^{16} + \alpha^{100}x^{15} + \alpha^{122}x^{14} + \alpha^{100}x^{13} + \alpha^{122}x^{12} + \alpha^{100}x^{11} + \alpha^{122}x^{10}.$$

och

$$g(x) = x^{10} + \alpha^{251}x^{9} + \alpha^{67}x^{8} + \alpha^{46}x^{7} + \alpha^{61}x^{6} + \alpha^{118}x^{5} + \alpha^{70}x^{4} + \alpha^{64}x^{3} + \alpha^{94}x^{2} + \alpha^{32}x + \alpha^{45}$$

För att båda polynomen ska ha samma högstagradstermen behöver $g(x)$ multipliceras med $\alpha^{6}x^{15}$. Det ger oss

$$\alpha^{6}x^{25} + \alpha^{2}x^{24} + \alpha^{73}x^{23} + \alpha^{52}x^{22} + \alpha^{67}x^{21} + \alpha^{124}x^{20} + \alpha^{76}x^{19} + \alpha^{70}x^{18} + \alpha^{100}x^{17} + \alpha^{38}x^{16} + \alpha^{51}x^{15}.$$

Vi subtraherar sedan resultatet från täljarpolynomet och får

$$(\alpha^{6} - \alpha^{6})x^{25} + (\alpha^{143} - \alpha^{2})x^{24} + (\alpha^{239} - \alpha^{73})x^{23} + (\alpha^{118} - \alpha^{52})x^{22} + (\alpha^{126} - \alpha^{67})x^{21} + (\alpha^{180} - \alpha^{124})x^{20} + (\alpha^{203} - \alpha^{76})x^{19} + (\alpha^{122} - \alpha^{70})x^{18} + (\alpha^{100} - \alpha^{100})x^{17} + (\alpha^{122} - \alpha^{38})x^{16} + (\alpha^{100} - \alpha^{51})x^{15} + (\alpha^{122} - 0)x^{14} + (\alpha^{100} - 0)x^{13} + (\alpha^{122} - 0)x^{12} + (\alpha^{100} - 0)x^{11} + (\alpha^{122} - 0)x^{10}$$

[6]För den som inte kommer ihåg algoritmen för polynomdivision, så görs det genom att multiplicera nämnarpolynomet med något sådant att både nämnarpolynomet och täljarpolynomet har samma högstagradsterm, sedan subtrahera dessa från varandra för att eliminera högstagradstermen. Detta görs stegvis till dess att det som är kvar har lägre grad än $g(x)$. Kom också ihåg att vi bara är intresserade av restpolynomet.

som ger oss

$$\alpha^{54}x^{24} + \alpha^{187}x^{23} + \alpha^{82}x^{22} + \alpha^{149}x^{21} + \alpha^{0}x^{20} + \alpha^{88}x^{19} + \alpha^{211}x^{18} + 0x^{17} + \alpha^{78}x^{16} + \alpha^{248}x^{15} + \alpha^{122}x^{14} + \alpha^{100}x^{13} + \alpha^{122}x^{12} + \alpha^{100}x^{11} + \alpha^{122}x^{10}. \quad (1)$$

Vi multiplicerar nu generatorpolynomet $g(x)$ med $\alpha^{54}x^{14}$ och får

$$\alpha^{54}x^{24} + \alpha^{50}x^{23} + \alpha^{121}x^{22} + \alpha^{100}x^{21} + \alpha^{115}x^{20} + \alpha^{172}x^{19} + \alpha^{124}x^{18} + \alpha^{118}x^{17} + \alpha^{148}x^{16} + \alpha^{86}x^{15} + \alpha^{99}x^{14}$$

som vi sedan subtraherar från (1) och får

$$(\alpha^{54} - \alpha^{54})x^{24} + (\alpha^{187} - \alpha^{50})x^{23} + (\alpha^{82} - \alpha^{121})x^{22} + (\alpha^{149} - \alpha^{100})x^{21} + (\alpha^{0} - \alpha^{115})x^{20} + (\alpha^{88} - \alpha^{172})x^{19} + (\alpha^{211} - \alpha^{124})x^{18} + (0 - \alpha^{118})x^{17} + (\alpha^{78} - \alpha^{148})x^{16} + (\alpha^{248} - \alpha^{86})x^{15} + (\alpha^{122} - \alpha^{99})x^{14} + (\alpha^{100} - 0)x^{13} + (\alpha^{122} - 0)x^{12} + (\alpha^{100} - 0)x^{11} + (\alpha^{122} - 0)x^{10}$$

som slutligen blir

$$\alpha^{96}x^{23} + \alpha^{188}x^{22} + \alpha^{42}x^{21} + \alpha^{243}x^{20} + \alpha^{128}x^{19} + \alpha^{36}x^{18} + \alpha^{118}x^{17} + \alpha^{142}x^{16} + \alpha^{151}x^{15} + \alpha^{40}x^{14} + \alpha^{100}x^{13} + \alpha^{122}x^{12} + \alpha^{100}x^{11} + \alpha^{122}x^{10}. \quad (2)$$

För att högstagradstermen i $g(x)$ ska bli densamma som (2) måste vi multiplicera med $\alpha^{96}x^{13}$, så vi gör det och får

$$\alpha^{96}x^{23} + \alpha^{92}x^{22} + \alpha^{163}x^{21} + \alpha^{142}x^{20} + \alpha^{157}x^{19} + \alpha^{214}x^{18} + \alpha^{156}x^{17} + \alpha^{160}x^{16} + \alpha^{190}x^{15} + \alpha^{128}x^{14} + \alpha^{141}x^{13}$$

som vi sedan subtraherar från (2), vilket blir

$$(\alpha^{96} - \alpha^{96})x^{23} + (\alpha^{188} - \alpha^{92})x^{22} + (\alpha^{149} - \alpha^{163})x^{21} + (\alpha^{243} - \alpha^{142})x^{20} + (\alpha^{128} - \alpha^{157})x^{19} + (\alpha^{36} - \alpha^{214})x^{18} + (\alpha^{118} - \alpha^{156})x^{17} + (\alpha^{142} - \alpha^{160})x^{16} + (\alpha^{151} - \alpha^{190})x^{15} + (\alpha^{30} - \alpha^{128})x^{14} + (\alpha^{100} - \alpha^{141})x^{13} + (\alpha^{122} - 0)x^{12} + (\alpha^{100} - 0)x^{11} + (\alpha^{122} - 0)x^{10}$$

som slutligen kan skrivas om som

$$\alpha^{88}x^{22} + \alpha^{85}x^{21} + \alpha^{189}x^{20} + \alpha^{54}x^{19} + \alpha^{187}x^{18} + \alpha^{116}x^{17} + \alpha^{127}x^{16} + \alpha^{2}x^{15} + \alpha^{215}x^{14} + \alpha^{2}x^{13} + \alpha^{122}x^{12} + \alpha^{100}x^{11} + \alpha^{122}x^{10}. \quad (3)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (3) måste vi multiplicera med $\alpha^{88}x^{12}$, så vi gör det och får

$$\alpha^{88}x^{22} + \alpha^{84}x^{21} + \alpha^{155}x^{20} + \alpha^{134}x^{19} + \alpha^{149}x^{18} + \alpha^{206}x^{17} + \alpha^{158}x^{16} + \alpha^{152}x^{15} + \alpha^{182}x^{14} + \alpha^{120}x^{13} + \alpha^{133}x^{12}$$

som vi sedan subtraherar från (3), vilket blir

$$(\alpha^{88} - \alpha^{88})x^{22} + (\alpha^{85} - \alpha^{84})x^{21} + (\alpha^{189} - \alpha^{155})x^{20} + (\alpha^{54} - \alpha^{134})x^{19} + (\alpha^{187} - \alpha^{149})x^{18} + (\alpha^{116} - \alpha^{206})x^{17} + (\alpha^{127} - \alpha^{158})x^{16} + (\alpha^{2} - \alpha^{152})x^{15} + (\alpha^{215} - \alpha^{182})x^{14} + (\alpha^{2} - \alpha^{133})x^{13} + (\alpha^{122} - \alpha^{133})x^{12} + (\alpha^{100} - 0)x^{11} + (\alpha^{122} - 0)x^{10}$$

som slutligen kan skrivas om som

$$\alpha^{109}x^{21} + \alpha^{36}x^{20} + \alpha^{222}x^{19} + \alpha^{78}x^{18} + \alpha^{178}x^{17} + \alpha^{172}x^{16} + \alpha^{145}x^{15} + \alpha^{197}x^{14} + \alpha^{166}x^{13} + \alpha^{112}x^{12} + \alpha^{100}x^{11} + \alpha^{122}x^{10}. \quad (4)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (4) måste vi multiplicera med $\alpha^{109}x^{11}$, så vi gör det och får

$$\alpha^{109}x^{21} + \alpha^{105}x^{20} + \alpha^{176}x^{19} + \alpha^{155}x^{18} + \alpha^{170}x^{17} + \alpha^{227}x^{16} + \alpha^{179}x^{15} + \alpha^{173}x^{14} + \alpha^{203}x^{13} + \alpha^{141}x^{12} + \alpha^{154}x^{11}$$

som vi sedan subtraherar från (4), vilket blir

$$(\alpha^{109} - \alpha^{109})x^{21} + (\alpha^{36} - \alpha^{105})x^{20} + (\alpha^{222} - \alpha^{176})x^{19} + (\alpha^{78} - \alpha^{155})x^{18} + (\alpha^{178} - \alpha^{170})x^{17} + (\alpha^{172} - \alpha^{227})x^{16} + (\alpha^{145} - \alpha^{179})x^{15} + (\alpha^{197} - \alpha^{173})x^{14} + (\alpha^{166} - \alpha^{203})x^{13} + (\alpha^{112} - \alpha^{141})x^{12} + (\alpha^{100} - \alpha^{154})x^{11} + (\alpha^{122} - 0)x^{10}$$

som slutligen kan skrivas om som

$$\alpha^{166}x^{20} + \alpha^{58}x^{19} + \alpha^{51}x^{18} + \alpha^{115}x^{17} + \alpha^{235}x^{16} + \alpha^{26}x^{15} + \alpha^{172}x^{14} + \alpha^{90}x^{13} + \alpha^{38}x^{12} + \alpha^{53}x^{11} + \alpha^{122}x^{10}. \quad (5)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (5) måste vi multiplicera med $\alpha^{166}x^{10}$, så vi gör det och får

$$\alpha^{166}x^{20} + \alpha^{162}x^{19} + \alpha^{233}x^{18} + \alpha^{212}x^{17} + \alpha^{227}x^{16} + \alpha^{29}x^{15} + \alpha^{236}x^{14} + \alpha^{230}x^{13} + \alpha^{5}x^{12} + \alpha^{198}x^{11} + \alpha^{211}x^{10}$$

som vi sedan subtraherar från (5), vilket blir

$$(\alpha^{166} - \alpha^{166})x^{20} + (\alpha^{58} - \alpha^{162})x^{19} + (\alpha^{51} - \alpha^{233})x^{18} + (\alpha^{115} - \alpha^{212})x^{17} + (\alpha^{235} - \alpha^{227})x^{16} + (\alpha^{26} - \alpha^{29})x^{15} + (\alpha^{172} - \alpha^{236})x^{14} + (\alpha^{90} - \alpha^{230})x^{13} + (\alpha^{38} - \alpha^{5})x^{12} + (\alpha^{53} - \alpha^{198})x^{11} + (\alpha^{122} - \alpha^{211})x^{10}$$

som slutligen kan skrivas om som

$$\alpha^{85}x^{19} + \alpha^{214}x^{18} + \alpha^{50}x^{17} + \alpha^{172}x^{16} + \alpha^{249}x^{15} + \alpha^{242}x^{14} + \alpha^{218}x^{13} + \alpha^{20}x^{12} + \alpha^{69}x^{11} + \alpha^{70}x^{10}. \quad (6)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (6) måste vi multiplicera med $\alpha^{85}x^{9}$, så vi gör det och får

$$\alpha^{85}x^{19} + \alpha^{81}x^{18} + \alpha^{152}x^{17} + \alpha^{131}x^{16} + \alpha^{146}x^{15} + \alpha^{203}x^{14} + \alpha^{155}x^{13} + \alpha^{149}x^{12} + \alpha^{179}x^{11} + \alpha^{117}x^{10} + \alpha^{130}x^{9}$$

som vi sedan subtraherar från (6), vilket blir

$$(\alpha^{85} - \alpha^{85})x^{19} + (\alpha^{214} - \alpha^{81})x^{18} + (\alpha^{50} - \alpha^{152})x^{17} + (\alpha^{172} - \alpha^{131})x^{16} + (\alpha^{249} - \alpha^{146})x^{15} + (\alpha^{242} - \alpha^{203})x^{14} + (\alpha^{218} - \alpha^{155})x^{13} + (\alpha^{20} - \alpha^{149})x^{12} + (\alpha^{69} - \alpha^{179})x^{11} + (\alpha^{170} - \alpha^{117})x^{10} - \alpha^{139}x^{9}$$

som slutligen kan skrivas om som

$$\alpha^{76}x^{18} + \alpha^{16}x^{17} + \alpha^{33}x^{16} + \alpha^{220}x^{15} + \alpha^{54}x^{14} + \alpha^{210}x^{13} + \alpha^{4}x^{12} + \alpha^{195}x^{11} + \alpha^{171}x^{10} + \alpha^{130}x^{9}. \quad (7)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (7) måste vi multiplicera med $\alpha^{76}x^8$, så vi gör det och får

$$\alpha^{76}x^{18} + \alpha^{72}x^{17} + \alpha^{143}x^{16} + \alpha^{122}x^{15} + \alpha^{137}x^{14} + \alpha^{194}x^{13} + \alpha^{146}x^{12} + \alpha^{140}x^{11} + \alpha^{170}x^{10} + \alpha^{109}x^{9} + \alpha^{121}x^{8}$$

som vi sedan subtraherar från (7), vilket blir

$$(\alpha^{76} - \alpha^{76})x^{18} + (\alpha^{16} - \alpha^{72})x^{17} + (\alpha^{33} - \alpha^{143})x^{16} + (\alpha^{220} - \alpha^{122})x^{15} + (\alpha^{54} - \alpha^{137})x^{14} + (\alpha^{210} - \alpha^{194})x^{13} + (\alpha^{4} - \alpha^{146})x^{12} + (\alpha^{195} - \alpha^{140})x^{11} + (\alpha^{171} - \alpha^{170})x^{10} - (\alpha^{130} - \alpha^{109})x^{9} - \alpha^{121}x^{8}$$

som slutligen kan skrivas om som

$$\alpha^{147}x^{17} + \alpha^{159}x^{16} + \alpha^{6}x^{15} + \alpha^{111}x^{14} + \alpha^{84}x^{13} + \alpha^{222}x^{12} + \alpha^{203}x^{11} + \alpha^{195}x^{10} + \alpha^{88}x^{9} + \alpha^{121}x^{8}. \quad (8)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (8) måste vi multiplicera med $\alpha^{147}x^7$, så vi gör det och får

$$\alpha^{147}x^{17} + \alpha^{143}x^{16} + \alpha^{214}x^{15} + \alpha^{193}x^{14} + \alpha^{208}x^{13} + \alpha^{10}x^{12} + \alpha^{217}x^{11} + \alpha^{211}x^{10} + \alpha^{241}x^{9} + \alpha^{179}x^{8} + \alpha^{192}x^{7}$$

som vi sedan subtraherar från (8), vilket blir

$$(\alpha^{147} - \alpha^{147})x^{17} + (\alpha^{159} - \alpha^{143})x^{16} + (\alpha^{6} - \alpha^{214})x^{15} + (\alpha^{111} - \alpha^{193})x^{14} + (\alpha^{84} - \alpha^{208})x^{13} + (\alpha^{222} - \alpha^{10})x^{12} + (\alpha^{203} - \alpha^{217})x^{11} + (\alpha^{195} - \alpha^{211})x^{10} + (\alpha^{68} - \alpha^{241})x^{9} + (\alpha^{121} - \alpha^{179})x^{8}) - \alpha^{192}x^{7}$$

som slutligen kan skrivas om som

$$\alpha^{33}x^{16} + \alpha^{60}x^{15} + \alpha^{170}x^{14} + \alpha^{9}x^{13} + \alpha^{88}x^{12} + \alpha^{172}x^{11} + \alpha^{85}x^{10} + \alpha^{207}x^{9} + \alpha^{228}x^{8} + \alpha^{192}x^{7}. \quad (9)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (9) måste vi multiplicera med $\alpha^{33}x^6$, så vi gör det och får

$$\alpha^{33}x^{16} + \alpha^{29}x^{15} + \alpha^{100}x^{14} + \alpha^{79}x^{13} + \alpha^{94}x^{12} + \alpha^{151}x^{11} + \alpha^{103}x^{10} + \alpha^{97}x^{9} + \alpha^{127}x^{8} + \alpha^{65}x^{7} + \alpha^{78}x^{6}$$

som vi sedan subtraherar från (9), vilket blir

$$(\alpha^{33} - \alpha^{33})x^{16} + (\alpha^{60} - \alpha^{29})x^{15} + (\alpha^{170} - \alpha^{100})x^{14} + (\alpha^{9} - \alpha^{79})x^{13} + (\alpha^{88} - \alpha^{94})x^{12} + (\alpha^{172} - \alpha^{151})x^{11} + (\alpha^{85} - \alpha^{103})x^{10} + (\alpha^{207} - \alpha^{97})x^{9} + (\alpha^{228} - \alpha^{127})x^{8} + (\alpha^{192} - \alpha^{65})x^{7} - \alpha^{78}x^{6}$$

som slutligen kan skrivas om som

$$\alpha^{74}x^{15} + \alpha^{164}x^{14} + \alpha^{73}x^{13} + \alpha^{24}x^{12} + \alpha^{161}x^{11} + \alpha^{70}x^{10} + \alpha^{223}x^{9} + \alpha^{174}x^{8} + \alpha^{77}x^{7} + \alpha^{78}x^{6}. \quad (10)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (10) måste vi multiplicera med $\alpha^{33}x^5$, så vi gör det och får

$$\alpha^{74}x^{15} + \alpha^{70}x^{14} + \alpha^{141}x^{13} + \alpha^{120}x^{12} + \alpha^{135}x^{11} + \alpha^{192}x^{10} + \alpha^{144}x^{9} + \alpha^{138}x^{8} + \alpha^{168}x^{7} + \alpha^{106}x^{6} + \alpha^{119}x^{5}$$

som vi sedan subtraherar från (10), vilket blir

$$(\alpha^{74} - \alpha^{74})x^{15} + (\alpha^{164} - \alpha^{70})x^{14} + (\alpha^{73} - \alpha^{141})x^{13} + (\alpha^{24} - \alpha^{120})x^{12} + (\alpha^{161} - \alpha^{135})x^{11} + (\alpha^{70} - \alpha^{192})x^{10} + (\alpha^{223} - \alpha^{144})x^{9} + (\alpha^{174} - \alpha^{138})x^{8} + (\alpha^{77} - \alpha^{168})x^{7} + (\alpha^{78} - \alpha^{106})x^{6} - \alpha^{119}x^{5}$$

som slutligen kan skrivas om som

$$\alpha^{17}x^{14} + \alpha^{90}x^{13} + \alpha^{20}x^{12} + \alpha^{78}x^{11} + \alpha^{187}x^{10} + \alpha^{63}x^9 + \alpha^{108}x^8 + \alpha^{31}x^7 + \alpha^{16}x^6 + \alpha^{119}x^5. \quad (11)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (11) måste vi multiplicera med $\alpha^{17}x^4$, så vi gör det och får

$$\alpha^{17}x^{14} + \alpha^{13}x^{13} + \alpha^{84}x^{12} + \alpha^{63}x^{11} + \alpha^{78}x^{10} + \alpha^{135}x^9 + \alpha^{87}x^8 + \alpha^{81}x^7 + \alpha^{111}x^6 + \alpha^{49}x^5 + \alpha^{62}x^4$$

som vi sedan subtraherar från (11), vilket blir

$$(\alpha^{17} - \alpha^{17})x^{14} + (\alpha^{90} - \alpha^{13})x^{13} + (\alpha^{20} - \alpha^{84})x^{12} + (\alpha^{78} - \alpha^{63})x^{11} + (\alpha^{187} - \alpha^{78})x^{10} + (\alpha^{63} - \alpha^{135})x^9 + (\alpha^{108} - \alpha^{87})x^8 + (\alpha^{31} - \alpha^{81})x^7 + (\alpha^{16} - \alpha^{111})x^6 + (\alpha^{119} - \alpha^{49})x^5 - \alpha^{62}x^4$$

som slutligen kan skrivas om som

$$\alpha^{241}x^{13} + \alpha^{90}x^{12} + \alpha^{96}x^{11} + \alpha^{149}x^{10} + \alpha^{3}x^9 + \alpha^{97}x^8 + \alpha^{33}x^7 + \alpha^{192}x^6 + \alpha^{113}x^5 + \alpha^{62}x^4. \quad (12)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (12) måste vi multiplicera med $\alpha^{241}x^3$, så vi gör det och får

$$\alpha^{241}x^{13} + \alpha^{237}x^{12} + \alpha^{53}x^{11} + \alpha^{32}x^{10} + \alpha^{47}x^9 + \alpha^{104}x^8 + \alpha^{56}x^7 + \alpha^{50}x^6 + \alpha^{80}x^5 + \alpha^{18}x^4 + \alpha^{31}x^3$$

som vi sedan subtraherar från (12), vilket blir

$$(\alpha^{241} - \alpha^{241})x^{13} + (\alpha^{90} - \alpha^{237})x^{12} + (\alpha^{96} - \alpha^{53})x^{11} + (\alpha^{149} - \alpha^{32})x^{10} + (\alpha^{3} - \alpha^{47})x^9 + (\alpha^{97} - \alpha^{104})x^8 + (\alpha^{33} - \alpha^{56})x^7 + (\alpha^{192} - \alpha^{50})x^6 + (\alpha^{113} - \alpha^{80})x^5 + (\alpha^{62} - \alpha^{18})x^4 - \alpha^{31}x^3$$

som slutligen kan skrivas om som

$$\alpha^{143}x^{12} + \alpha^{174}x^{11} + \alpha^{154}x^{10} + \alpha^{218}x^9 + \alpha^{209}x^8 + \alpha^{229}x^7 + \alpha^{13}x^6 + \alpha^{95}x^5 + \alpha^{233}x^4 + \alpha^{31}x^3. \quad (13)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (13) måste vi multiplicera med $\alpha^{143}x^2$, så vi gör det och får

$$\alpha^{143}x^{12} + \alpha^{139}x^{11} + \alpha^{210}x^{10} + \alpha^{189}x^9 + \alpha^{204}x^8 + \alpha^{6}x^7 + \alpha^{213}x^6 + \alpha^{207}x^5 + \alpha^{237}x^4 + \alpha^{175}x^3 + \alpha^{188}x^2$$

som vi sedan subtraherar från (13), vilket blir

$$(\alpha^{143} - \alpha^{143})x^{12} + (\alpha^{174} - \alpha^{139})x^{11} + (\alpha^{154} - \alpha^{210})x^{10} + (\alpha^{218} - \alpha^{189})x^9 + (\alpha^{209} - \alpha^{204})x^8 + (\alpha^{229} - \alpha^{6})x^7 + (\alpha^{13} - \alpha^{213})x^6 + (\alpha^{95} - \alpha^{207})x^5 + (\alpha^{233} - \alpha^{237})x^4 + (\alpha^{31} - \alpha^{175})x^3 - \alpha^{188}x^2$$

som slutligen kan skrivas om som

$$\alpha^{171}x^{11} + \alpha^{30}x^{10} + \alpha^{115}x^9 + \alpha^{87}x^8 + \alpha^{9}x^7 + \alpha^{21}x^6 + \alpha^{102}x^5 + \alpha^{78}x^4 + \alpha^{166}x^3 + \alpha^{188}x^2. \quad (14)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (14) måste vi multiplicera med $\alpha^{171}x$, så vi gör det och får

$$\alpha^{171}x^{11} + \alpha^{167}x^{10} + \alpha^{238}x^9 + \alpha^{217}x^8 + \alpha^{232}x^7 + \alpha^{34}x^6 + \alpha^{241}x^5 + \alpha^{235}x^4 + \alpha^{10}x^3 + \alpha^{203}x^2 + \alpha^{216}x$$

som vi sedan subtraherar från (14), vilket blir

$$(\alpha^{171} - \alpha^{171})x^{11} + (\alpha^{30} - \alpha^{167})x^{10} + (\alpha^{115} - \alpha^{238})x^9 + (\alpha^{87} - \alpha^{217})x^8 + (\alpha^{9} - \alpha^{232})x^7 + (\alpha^{21} - \alpha^{34})x^6 + (\alpha^{102} - \alpha^{241})x^5 + (\alpha^{78} - \alpha^{235})x^4 + (\alpha^{166} - \alpha^{10})x^3 + (\alpha^{188} - \alpha^{203})x^2 - \alpha^{216}x$$

som slutligen kan skrivas om som

$$\alpha^{76}x^{10} + \alpha^{43}x^9 + \alpha^{156}x^8 + \alpha^{12}x^7 + \alpha^{120}x^6 + \alpha^{200}x^5 + \alpha^{119}x^4 + \alpha^{179}x^3 + \alpha^{221}x^2 + \alpha^{216}x. \quad (15)$$

För att högstagradstermen i $g(x)$ ska bli densamma som i (15) måste vi multiplicera med α^{76}, så vi gör det och får

$$\alpha^{76}x^{10} + \alpha^{72}x^9 + \alpha^{143}x^8 + \alpha^{122}x^7 + \alpha^{137}x^6 + \alpha^{194}x^5 + \alpha^{146}x^4 + \alpha^{140}x^3 + \alpha^{170}x^2 + \alpha^{108}x^1 + \alpha^{121}$$

som vi sedan subtraherar från (15), vilket blir

$$(\alpha^{76} - \alpha^{76})x^{10} + (\alpha^{43} - \alpha^{72})x^9 + (\alpha^{156} - \alpha^{143})x^8 + (\alpha^{12} - \alpha^{122})x^7 + (\alpha^{120} - \alpha^{137})x^6 + (\alpha^{200} - \alpha^{194})x^5 + (\alpha^{119} - \alpha^{146})x^4 + (\alpha^{179} - \alpha^{140})x^3 + (\alpha^{221} - \alpha^{170})x^2 + (\alpha^{216} - \alpha^{108})x^1 - \alpha^{121}$$

som kan skrivas om som

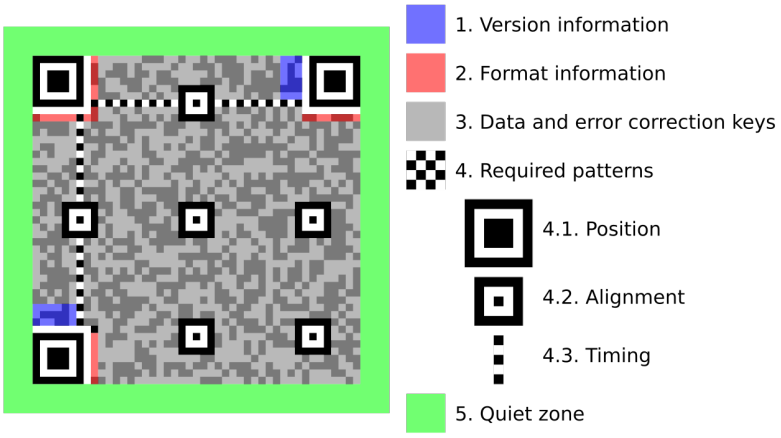
$$\alpha^{224}x^9 + \alpha^{242}x^8 + \alpha^{138}x^7 + \alpha^{188}x^6 + \alpha^{130}x^5 + \alpha^{223}x^4 + \alpha^{246}x^3 + \alpha^{153}x^2 + \alpha^{14}x + \alpha^{121}. \quad (16)$$

Polynomet ovan har en grad lägre än generatorpolynomet, alltså är det precis det vi letar efter. Vi skriver om till decimalform, som ger oss att felrättningsbiten kommer att vara $(18, 176, 33, 165, 46, 9, 207, 146, 19, 118)_{10}$.
Det ger oss att vår datasträng nu är
$(64, 84, 22, 199, 102, 150, 224, 236, 17, 236, 17, 236, 17, 236, 17, 236, 18, 176, 33, 165, 46, 9, 207, 146, 19, 118)_{10}$.

4.4 Fasta mönster

När man med blotta ögat tittar på en QR-kod är det lätt att ögonen dras till dess mest utmärkande särdrag, nämligen de fasta mönstren. De fasta mönstren spelar en viktig roll i hur en QR-kod skannas, och ser därför likadana ut för alla QR-koder, oavsett vilken data koden försöker framföra. De flesta har nog lagt märke till åtminstone en av dessa, nämligen de tre rutorna i det övre högra hörnet, övre

vänstra hörnet och nedre vänstra hörnet. De kallas positionsmarkörer och har som syfte att hjälpa maskinen som ska läsa av QR-koden att orientera sig, så att koden avläses från rätt håll. Vi vet att vi ska börja från det nedre högra hörnet, eftersom det inte finns någon positionsmarkör där.
En QR-kod har också timingmönster, som alltid befinner sig på rad och kolumn[7] 6, dock inte så att de överlappar med positionsmarkörerna. Timingmönstret består av alternerande svarta och vita pixlar. Deras syfte är att hjälpa skannern, som åtminstone i QR-kodens tidiga historia läste av koden med hjälp av en laser som färdades över koden, att skanna i rätt hastighet.
För större QR-koder behövs även inriktningsmönster. Dessa påminner om positionsmarkörerna, men är lite mindre, och befinner sig mitt i koden snarare än i hörnen. De kan också vara fler än tre. Deras uppgift är att se till att QR-koden skannas korrekt, även om den är böjd eller förvrängd på något annat sätt.
Till sist måste det runt QR-koden vara enfärgat, för att skannern inte ska förvirras och tro att bakgrunden är en del av koden. [Gar15]



Figur 1: De olika fasta mönstren i en QR-kod(CC BY-SA 3.0) [Bob13]

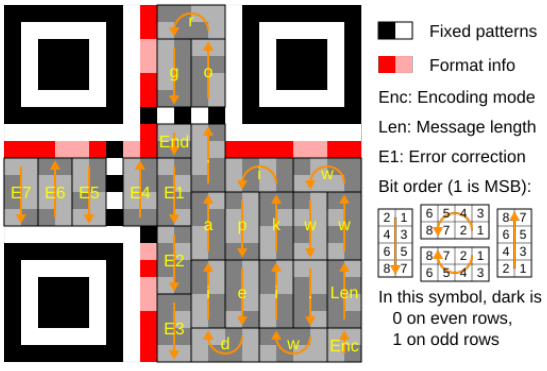
Eftersom de fasta mönstren ser likadana ut varje gång behövs ingen komplicerad uträkning. Det är bara att rita!

[7]Både rad och kolumnnummer börjar från 0.

Figur 2: Efter att vi ritat ut våra fasta mönster ser koden ut såhär. Det röda fältet är reserverat för formatinformationen, som vi diskuterar i ett framtida kapitel.

4.5 Rita ut data

Nu är det dags att äntligen rita ut datan, inklusive felkorrigeringen, i koden. I de förra kapitlen har vi kommit fram till att vår datasträng är $(64, 84, 22, 199, 102, 150, 224, 236, 17, 236, 17, 236, 17, 236, 17, 236, 18, 176, 33, 165, 46, 9, 207, 146, 19, 118)_{10}$. QR-koder kan dock bara har svarta eller vita pixlar. Därför måste vi konvertera vår datasträng till binär. Vi får då $(01000000, 01010100, 00010110, 11000111, 01100110, 10010110, 11100000, 11101100, 00010001, 11101100, 00010001, 11101100, 00010001, 11101100, 00010001, 11101100, 00010010, 10110000, 00100001, 10100101, 00101110, 00001001, 11001111, 10010010, 00010011, 01110110)_2$.



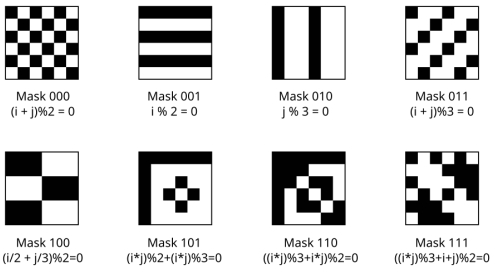
Figur 3: Här kan vi se hur datan ska placeras ut i ett sicksackmönster.(CC0)

Vi sätter in vår data i de grå fälten från figur 2 och vår QR-kod ser nu ut såhär:

Figur 4: QR-koden efter att datan utplacerats. Svart pixel är en etta, vit pixel en nolla. Röda fält är formatinformation.

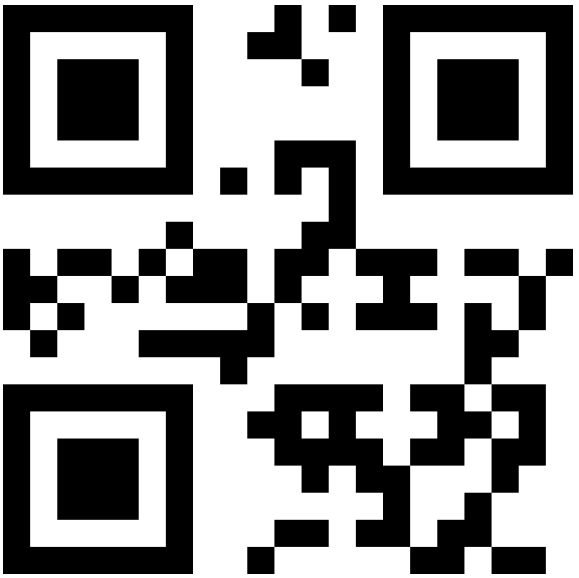
4.6 Att välja det bästa maskeringsmönstret

Maskeringsmönster för en QR-kod är en funktion som förbättrar läsbarheten och träffsäkerheten vid skanning, genom att lägga på ett överlägg. Vår QR-kod från det förra delkapitlet hade nog gått att läsa av, men med vissa svårigheter och större risk för fel. En skanner kan exempelvis ha lite svårt för stora enfärgade fält. Därför vill vi invertera vissa pixlar för att skapa en jämnare fördelning av svart och vitt.



Figur 5: De 8 maskeringsmönstren. I bilden motsvarar i rad och j kolumn. Dessa läggs sedan ovan på QR-koden(endast datadelen), och pixlar som hamnar under de svarta delarna av maskeringsmönstret inverteras.

För närvarande ser vår QR-kod ut så här.



Figur 6: De röda bitarna som reserverats för formatinformation har, för den här processen, färgats vita.

Vi bestämmer vilket maskeringsmönster som är bäst genom att testa lägga på alla, utvärdera resultatet på fyra olika kriterier genom att räkna ihop hur många straffpoäng varje maskeringsmönster fick, och sedan välja det maskeringsmönster som fick lägst straffpoäng. Kriterierna är som följande:

Kriterium 1: Sammanhängande rader och kolumner.
Om en sekvens i en rad eller en kolumn löper för länge utan att byta färg straffas maskeringsmönstret. 3 straffpoäng ifall 5 pixlar av samma färg hänger ihop, 4 poäng för ifall 6 pixlar av samma färg hänger ihop, 5 poäng ifall 7 pixlar hänger ihop, och så vidare. Sekvenserna kan inte överlappa med varandra.

Kriterium 2: Sammanhängade lådor.
För varje låda om 2×2 pixlar av samma färg straffas maskeringsmönstret med 3 straffpoäng. Lådorna kan överlappa med varandra.

Kriterium 3: Mönster som liknar positionmarkörerna.
För varje sekvens, horisontellt eller vertikalt, som består av pixlarna $4\times vit, svart, vit, 3\times svart, vit, svart$ antingen baklänges eller framlänges läggs 40 straffpoäng till. Sekvenser som liknar positionmarkörerna kan överlappa.

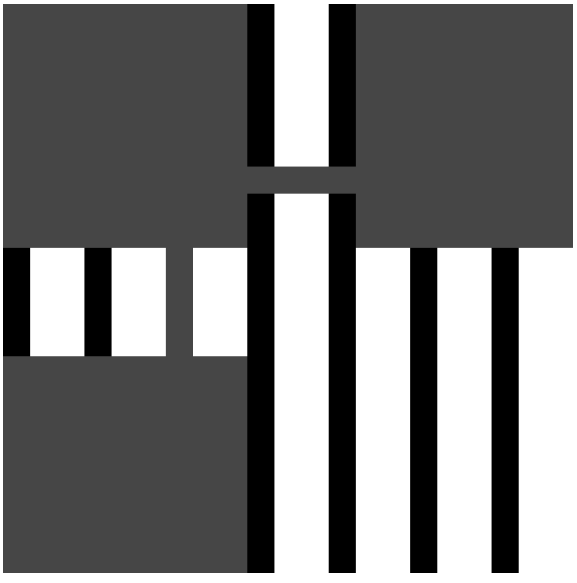
Kriterium 4: Jämn fördelning av färg.
Om andelen mörka pixlar är inom intervallet [45%,55%] tilldelas inga straffpoäng.

Om andelen mörka pixlar är inom intervallet [40%,60%] tilldelas 10 straffpoäng. Om andelen mörka pixlar är inom intervallet [35%,65%] tilldelas 20 straffpoäng, och så vidare.

Till sist summeras straffpoängen för varje kriterium, och det maskeringsmönster med lägst straffpoäng väljs. Notera att straffpoäng tilldelas även för de fasta mönstren, trots att maskeringsmönstret bara läggs över datadelen.

Maskeringsmönster	Kriterium 1	Kriterium 2	Kriterium 3	Kriterium 4	Summa
000	173	132	800	0	1105
001	191	120	960	0	1271
010	196	105	720	0	1021
011	183	138	880	0	1201
100	174	141	840	0	1155
101	183	126	880	0	1189
110	202	150	840	0	1192
111	183	135	840	0	1158

I vårt fall är maskeringsmönster 010 bäst, så vi lägger det över vår QR-kod.



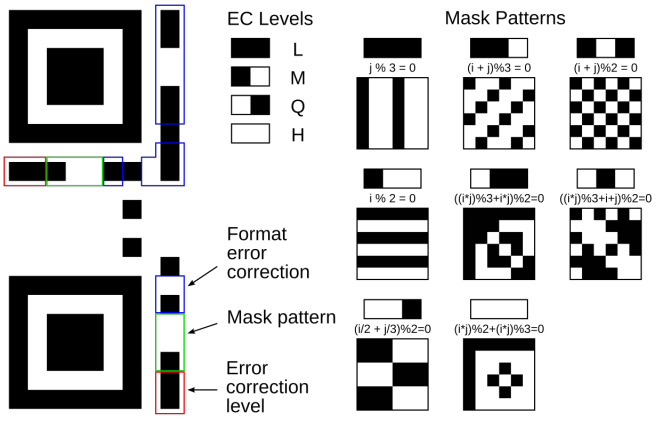
Figur 7: Maskeringsmönster 010 som läggs över vår QR-kod.

Vårt resultat ser ut så här.

Figur 8: Vår QR-kod med maskeringsmönster.

4.7 Rita in formatinformation

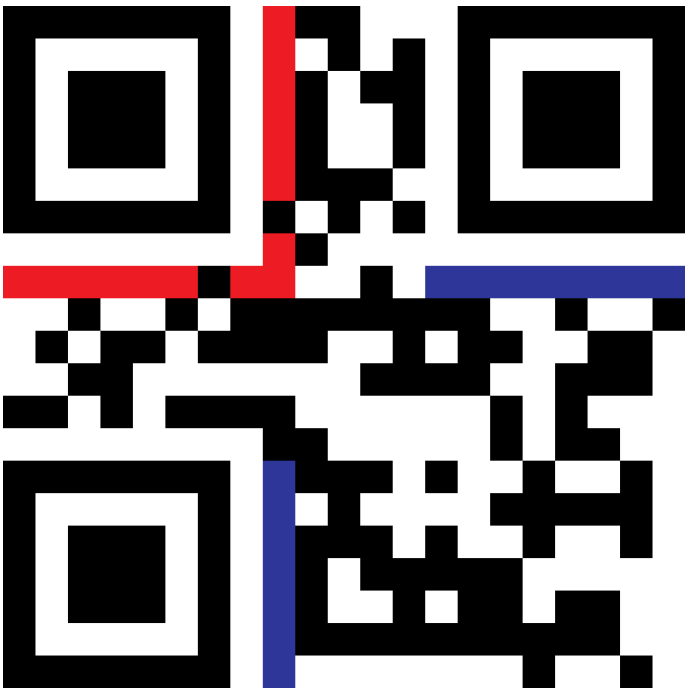
Formatinformationen innehåller information om vilken felrättningsnivå vi valt, och vilket maskeringsmönster vi valt, och finns där för att berätta för QR-kod skannern vad den ska göra med det den har skannat.



Figur 9: Som vi ser i bilden består formatinformationen av en sträng 1:or och 0:or av längd 5. På dessa läggs felkorrigering på så att vi får en sträng av total längd 15 med felkorrigeringen.

Formatinformationen ritas ut på två ställen, i figuren nedan utritade i rött och blått. Den som vill kan räkna att vi har 15 röda och 15 blå pixlar.

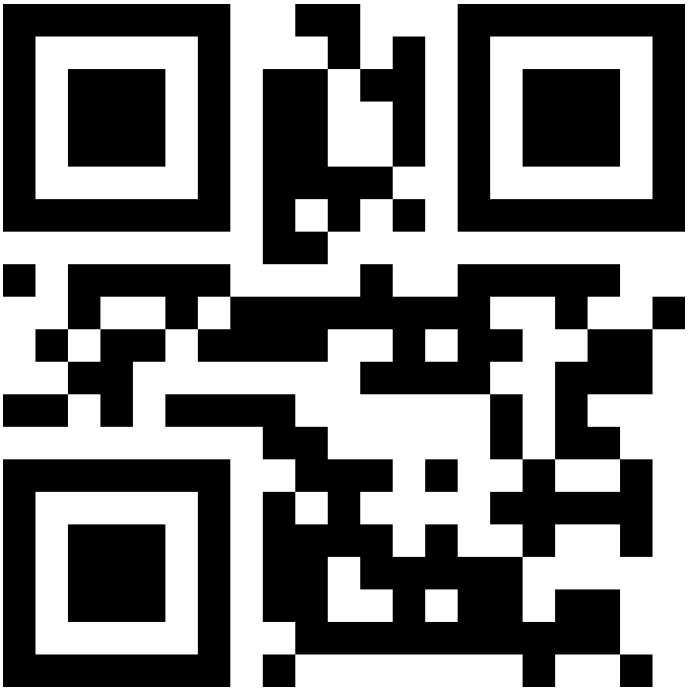
Felkorrigeringen för formatinformationen sköts på liknande vis som med datan, allt-



Figur 10: Den kodade formatinformationen placeras ut på två ställen. Det blå avläses först nedifrån och upp, sedan vänster till höger. Dett röda avläses först vänster till höger, sedan nedifrån och upp.

så med hjälp av Reed-Solomon-koder. I detta fall, eftersom vi bara jobbar med 1:or och 0:or, behöver vi dock inte konstruera någon ändlig kropp med hjälp av primitiva polynom. Vi har redan en, nämligen heltalen modulo 2. Dessutom är vårt generatorpolynom redan skapat åt oss, $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$. Vi antar att vår formatinformationssträng kan representeras av ett fjärdegradspolynom över heltalen modulo 2, och sedan kan vi tillämpa samma metod[8] som i avsnitt 4.3. Enda skillnaden är att, när vi är färdiga med Reed-Solomon-kodningen för formatinformationen ska resultatet adderas med polynomet $x^{14} + x^{12} + x^9 + x^5 + x^4 + x^3 + x^2 + 1$. Summan blir vår kodade formatinformation. I vårt fall får vi 101111001111100, som vi placerar på det röda och det blå fältet. Slutligen får vi vår QR-kod.

[8]Det går dock, eftersom det bara finns $2^5 = 32$ olika kombinationer, går det att för-generera alla, och sedan bara ha formatinformationen i en tabell. En sådan tabell går att hitta här.



Figur 11: Den fullbordade QR-koden. Prova skanna vet jag!

Referenser

[Big02] Norman Biggs. *Discrete mathematics*. Oxford Univ. Press, Oxford, 2. ed. edition, 2002.

[BKS21] Alexandre M. Bayen, Qingkai Kong, and Timmy Siauw. *Python programming and numerical methods: A guide for engineers and scientists*. Academic Press is an imprint of Elsevier, 2021.

[Bob13] Bobmath. Qr code structure example 3, 2013. File: QR Code Structure Example 3.svg. URL: https://commons.wikimedia.org/wiki/File:QR_Code_Structure_Example_3.svg#/media/File:QR_Code_Structure_Example_3.svg.

[Gar15] Gautam Garg. Qr code structure: Everything you need to know, Feb 2015. URL: https://scanova.io/blog/qr-code-structure/.

[MS04] Victor Chen Madhu Sudan. 6.895 essential coding theory, lecture 4. https://ocw.mit.edu/courses/6-895-essential-coding-theory-fall-2004/cce7cdbf647cce78b35aae128172671c_lect04.pdf, 2004. [Accessed 30-12-2024].

[Mul24] Derek Alexander Muller. I built a qr code with my bare hands to see how it works, Sep 2024. URL: https://www.youtube.com/watch?v=w5ebcowAJD8.

[Nay] Nayuki. Creating a qr code step by step -- nayuki.io. https://www.nayuki.io/page/creating-a-qr-code-step-by-step. [Accessed 22-12-2024].

[Tho22] Thonky. Thonky's qr code tutorial, Aug 2022. URL: https://www.thonky.com/qr-code-tutorial/.

[Ver22] Tom Verbeure. Reed-solomon error correcting codes from the bottom up, Aug 2022. URL: https://tomverbeure.github.io/2022/08/07/Reed-Solomon.html.

[Wik23] Wikipedia contributors. Systematic code -- Wikipedia, the free encyclopedia, 2023. [Online; accessed 8-December-2024].

URL: https://en.wikipedia.org/w/index.php?title=Systematic_code&oldid=1177668671.

[Wik24a] Wikipedia. Reed-solomon error correction -- wikipedia, the free encyclopedia, 2024. [Online; accessed 16-October-2024]. URL: https://simple.wikipedia.org/w/index.php?title=Reed%E2%80%93Solomon_error_correction&oldid=9473910.

[Wik24b] Wikipedia contributors. Primitive polynomial (field theory) -- Wikipedia, the free encyclopedia, 2024. [Online; accessed 30-December-2024]. URL: https://en.wikipedia.org/w/index.php?title=Primitive_polynomial_(field_theory)&oldid=1225650769.