Algebra behind the scenes : Reed-Solomon Algorithm

by Debashish Sharma - Saturday, April 29, 2017

http://gonitsora.com/algebra-behind-scenes-reed-solomon-algorithm/

You must have seen QR codes displayed in Paytm stickers. One can scan these codes to make payment to the merchant easily via Paytm. Such QR codes are used for making super fast transactions via the newly introduced BHIM app (Bharat Interface for Money) and UPI apps (Unified Payments Interface). The advantage of a QR code is that it has abolished the requirement of entering account details, IFS code, branch name etc. in order to carry out online money transfers. A QR code consists of black squares arranged in a square grid on a white background, which can be read by an imaging device such as a camera, and processed using Reed–Solomon error correction until the image can be appropriately interpreted. The required data is then extracted from patterns that are present in both horizontal and vertical components of the image. The Reed-Solomon error correction algorithm makes sure that even if the QR code is slightly damaged due to wear and tear, a certain percentage of error can be detected and the correct information hidden in the code can be extracted safely. This algorithm uses a lot of algebra. A poster presentation on this algebra was made by my students Rahul Paul and Rajarshee Rohan Suklabaidya of B.Sc 2nd Semester, Mathematics Honours, Gurucharan College, at the College week multi-disciplinary exhibition. Later, Rahul Paul presented an improved version of the presentation in the National Conference on Recent Trends in Mathematical Sciences at Assam University and was adjudged the best poster award. Here, I present a glimpse of the algebra behind the Reed Solomon codes.

The pre-requisites include an introduction to modular arithmetic and polynomials over finite fields, which are usually taught in the first year of an under-graduate course in mathematics in most Indian Universities. This methodology can be followed in explaining Reed-Solomon codes with the intention of improving the teaching-learning process in an abstract algebra class.

1. If $latex p$ is a prime number, then the set $latex \mathbb{Z}_p=\lbrace 0,1,2,\ldots,p-1 \rbrace$ together with the operations $latex +_p$ and $latex \times_p$ is a finite field. $latex a+_pb$ is the remainder when $latex a+b$ is divided by $latex p$. $latex a\times_pb$ is the remainder when $latex a\times b$ is divided by $latex p$.
2. An expression of the form $latex P(x)=a_0+a_1x+a_2x^2+\ldots+a_nx^n$ where $latex a_0,a_1,a_2,\ldots,a_n\in \mathbb{Z}_p$ and $latex a_n\neq 0$, is called a polynomial of degree $latex n$ over the finite field $latex \mathbb{Z}_p$. For example, $latex Q(x)=x+x^3+5x^5+2x^8$ is a polynomial over $latex \mathbb{Z}_7$.
3. Finite differences : The first forward difference of a sequence $latex a_1,a_2,a_3,\ldots$ is the sequence $latex \Delta a_i=a_{i+1}-a_i$. The second order difference is $latex \Delta^2 a_i=\Delta (\Delta a_i)=\Delta (a_{i+1}-a_i)=a_{i+2}-2a_{i+1}+a_i$. Similarly, $latex \Delta^3 a_i$, $latex \Delta^4 a_i$ etc. are defined.

Let $latex p$ be a prime number and let $latex m\leq n\leq p$. The Reed-Solomon code over the field $latex \mathbb{Z}_p$ with $latex m$ message symbols and $latex n$ code symbols is defined as : Given a message vector $latex \begin{pmatrix}a_1&a_2&a_3&\cdots & a_m\end{pmatrix}$, where the symbols are in $latex \mathbb{Z}_p$, let $latex P(x)$ be the polynomial $latex

P(x)=a_1+a_2x+a_3x^2+\cdots+a_{m-1}x^{m-2}+a_mx^{m-1}$ with coefficients given by the message symbols. Thus, $latex P(x)$ is a polynomial of degree at most $latex m-1$ in one variable $latex x$ with coefficients from $latex \mathbb{Z}_p$. Then, the code vector for this message is the list of the first $latex n$ values of the polynomial $latex P(x)$ evaluated using modular arithmetic in $latex \mathbb{Z}_p$. Let $latex m=3, n=7, p=7$ i.e. we are to create a code on $latex 7$ symbols for a message with $latex 3$ symbols over the finite field $latex \mathbb{Z}_7=\lbrace 0,1,2,3,4,5,6 \rbrace$. Let the message vector be $latex \begin{pmatrix}2&3&4\end{pmatrix}$. Then, the message polynomial is $latex P(x)=2+3x+4x^2$ and the code vector is computed as : $latex P(i)=2+3i+4i^2$ for $latex i=0,1,2,\cdots ,6$. Thus,

$latex P(0)=2+3\times 0+4\times 0^2=2$

$latex P(1)=2+3\times 1+4\times 1^2=9=2\text{ (mod 7)}$

and so on.

Thus, the message $latex \begin{pmatrix}2&3&4\end{pmatrix}$ is coded as $latex \begin{pmatrix}2&2&3&5&1&5&3\end{pmatrix}$$

The code is transmitted to the receiver and there is always a possibility of error in the transmission. Such errors are reflected in the entries of the received vector. For example,

Actual code : $latex \begin{pmatrix}2&2&3&5&1&5&3\end{pmatrix}$

Received code : $latex \begin{pmatrix}2&2&6&5&3&5&3\end{pmatrix}$

So, there are errors in the 3rd (i=2) and 5th (i=4) positions.

It has been shown that a Reed-Solomon code with $latex m$ message symbols and $latex n=m+2e$ code symbols can correct at most $latex e$ errors. Let the actual transmitted code vector be

$latex \begin{pmatrix}P(0)&P(1)&\cdots & P(n-1)\end{pmatrix}$

and the received vector be

$latex \begin{pmatrix}R_0&R_1&\cdots &R_{n-1}\end{pmatrix}$.

1. If there are no errors, then $latex R_i=P(i)$ for all $latex i=0,1,\cdots,n-1$.
2. If there are at most $latex e$ errors, then $latex R_i\neq P(i)$ for at most $latex e$ values of $latex i=0,1,2\cdots,n-1$.
3. In the above example, $latex R_2=6\neq P(2)=3$ and $latex R_4=3\neq P(4)=1$.
4. In reality, the error positions are unknown.

Let $latex i_1,i_2,\cdots,i_k$ be the error positions. Then, we construct the polynomial, say error polynomial, as

$latex E(x)=(x-i_1)(x-i_2)\cdots (x-i_k)$

which is of degree at most $latex k\leq e$. Thus, each error position is a zero of the polynomial $latex E(x)$. We consider the polynomial identity

$latex Q(x)=P(x)E(x)$ (Eq. (1))

Since $latex E(x)$ is of degree $latex \leq e$ and $latex P(x)$ is of degree $latex \leq m-1$, so $latex Q(x)$ is of degree $latex \leq m-1+e=m+e-1$. We take,

$latex E(x)=u_0+u_1x+u_2x^2+\cdots+u_{e}x^e$

$latex Q(x)=v_0+v_1x+v_2x^2+\cdots+v_{m+e-1}x^{m+e-1}$

where the coefficients $latex u_j$ and $latex v_j$ are unknowns. Thus, the total number of unknowns is $latex (e+1)+(m+e-1+1)=m+2e+1=n+1$. Now, from equation (1),

$latex Q(i)=P(i)E(i)\text{ for all }i=0,1,\ldots, n-1$ (Eq. (2))

Thus, $latex Q(i)=R_iE(i)$ if $latex i$ is not an error position and $latex Q(i)=P(i)\times 0$ if $latex i$ is an error position. Thus, in both cases, we have $latex Q(i)=R_iE(i) \text{ for all }i=0,1,\ldots, n-1$

1. The equation (2) is called the *key equation* for the decoding algorithm.
2. It is a system of $latex n$ linear equations in $latex n+1$ unknowns ($latex u_j$'s and $latex v_j$'s).
3. Solving this system of equations, we can get the expressions for $latex Q(x)$ and $latex E(x)$.
4. From eq. (1), the message polynomial $latex P(x)$ is obtained by dividing $latex Q(x)$ by $latex E(x)$.

Since degree of $latex Q(x)\leq m+e-1$, so its forward differences of order $latex m+e$ or more vanish. So, in particular

$latex \Delta^{m+e}(Q(i))=0\text{ for all }i=0,1,2,\ldots,e-1$

$latex \Rightarrow \Delta^{m+e}(R_iE(i))=0$

$latex \Rightarrow \Delta^{m+e}(u_0R_i+u_1iR_i+u_2i^2R_i+\ldots +u_ei^eR_i)=0$

$latex \Rightarrow u_0\Delta^{m+e}(R_i)+u_1\Delta^{m+e}(iR_i)+u_2\Delta^{m+e}(i^2R_i)+\ldots +u_e\Delta^{m+e}(i^eR_i)=0$

Thus, we obtain the following system of $latex e$ linear equations in $latex e+1$ unknowns :

$latex B\begin{pmatrix}u_0& u_1& \ldots & u_e\end{pmatrix}^T=0$

where the $latex (i,j)$th entry of the matrix $latex B$ is $latex b_{ij}=\Delta^{m+e}(i^jR_i)$, where

$latex i=0,1,\ldots,e-1$ and $latex j=0,1,\ldots,e$. This can be solved to obtain the unknowns $latex u_0,u_1,\ldots,u_e$. Hence, we get $latex E(x)$. Now, the values $latex Q(0), Q(1),\ldots, Q(n-1)$ are calculated and $latex Q(x)$ is obtained by Newton's forward interpolation formula :

$latex Q(x)=Q(0)+x\Delta Q(0)+\dfrac{x(x-1)}{2!}\Delta^2 Q(0)+\dfrac{x(x-1)(x-2)}{3!}\Delta^3Q(0)+\ldots$

The error polynomial $latex E(x)$ is found to be $latex E(x)=1+x+x^2$. The values of $latex Q(i)$ are computed as $latex 2,6,0,2,0,1,3$. Thus, the difference table is computed modulo 7 :

$latex \begin{matrix}&2&6&0&2&0&1&3\\\Delta &4&1&2&5&1&2&\\\Delta^2 &4&1&3&3&1& &\\\Delta^3 &4&2&0&5& & &\\\Delta^4 &5&5&5&&&&\end{matrix}$

Hence, by Newton's forward interpolation formula, we obtain

$latex Q(x)=2+4x+4\dfrac{x(x-1)}{2!}+4\dfrac{x(x-1)(x-2)}{3!}+5\dfrac{x(x-1)(x-2)(x-3)}{4!}$

$latex \Rightarrow Q(x)=2+5x+2x^2+4x^4$

And hence, $latex P(x)=\dfrac{Q(x)}{E(x)}=2+3x+4x^2$.

Thus, the original message has been detected as $latex \begin{pmatrix}2&3&4\end{pmatrix}$ even though there were two errors in the received code.

Error correction levels in QR codes

If the number of message symbols is $latex m$ and the Reed-Solomon code is required to correct the errors in at most $latex e$ symbols, then the number of symbols in the Reed-Solomon code has to be $latex n = m+2e$. The ratio $latex \dfrac{e}{n}$ gives the error correction level of the Reed-Solomon code. Expressed as percentage, the error correction level in the corresponding QR code is $latex \dfrac{e}{n}\times 100%$. This percentage is denoted in the QR code as under :

Error Correction Level indicators

Level L

Level M

Level Q

Level H

Thus, we see that there is a lot of mathematics is hidden behind these black and white drawings. This gives us a glimpse of the wonderful applications of abstract algebra.
