Algorithm Design Techniques

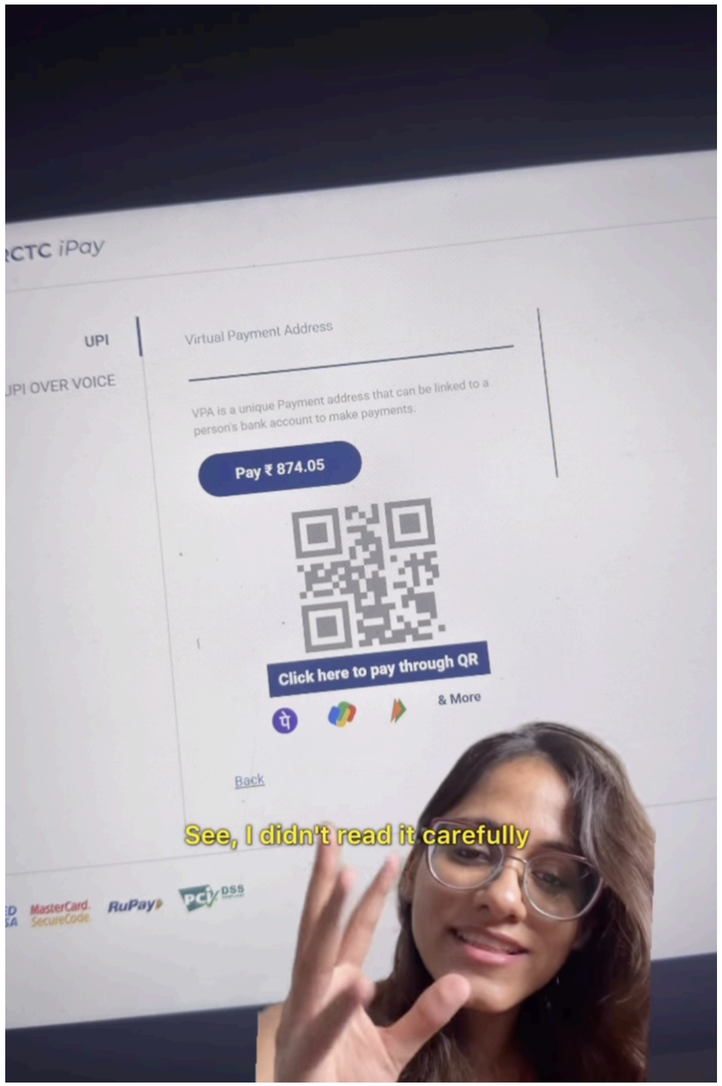
Tutorial 1: Introduction and Codes

Introduction

If you go to the IRCTC website (to book a train) and decide to pay by UPI, you will get the option to generate a QR code.

If you are sleepy as Ria Chopra, and you scan the placeholder code directly, you will get the phrase "'**twas brillig**"

If anyone knows the meaning/origin of this phrase, this is your opportunity to get a chocolate. 



See, QR codes are not just a payment utility. We can use them to link to any website or food menus or actually anything. You guys are old, but I am sure you must have seen QRs used in these places before. These things can store quite a bit of information. And are very damage resistant. Throw sauce at it, scan it in a moving vehicle or even erase parts of it, it will always work!

In this case, this is the first line of the poem "Jabberwoky" by Lewis Carrol which featured in "Through The Looking Glass" aka sequel to "Alice in Wonderland".

And ofcourse, we don't expect IRCTC people to do deep literature references. So what in the world was this?

Hello!

The TAs for this course are:

- **Arjun Maneesh Agarwal (me!!!)**
- **Harshitha Mucherla**
- **Vardhan Kumar Ray**

We will have the tutorials every **Friday** from **10:30 am**.

If you have doubts, feel free to interject during the tutorials or if and when you can find us (and we are free).

About Me

While I personally do all things algo, my interest broadly lies in using algorithmic techniques to model, predict and understand social and behavioural dynamics; and use it to make better mechanisms and better choices. Hence a lot of examples will be borrowed from such places. (Which I feel is a good thing as you all also care about using algorithms in real life).

Anyways, I yap a lot and overshare like anything so please feel free to cut me off when I start yapping or oversharing. Also, please feel free to give me feedback regarding anything.

Some of the class questions we will have an attached chocolate points and will get, well a chocolate.

Anyways, with that let's continue with algorithms.

What are Algorithms?

Before we get to back to QR codes and poems and all that, perhaps it's worth pausing briefly to question what 'algorithm' actually means. It's a term that, although used frequently, routinely fails to convey much actual information. This is partly because the word itself is quite vague. Officially, it is defined as follows:

algorithm (noun): A step-by-step procedure for solving a problem or accomplishing some end especially by a computer.

By this definition, a brownie recipe, self-help books, Youtube tutorials are all algorithms.

For this course, our focus will be on algorithms as mathematical objects. They still boil down to a list of step-by-step instructions but the steps are much more precise.

So what is our goal?

The study of algorithms tends to focus on designing algorithms which are optimal in some sense. We normally want them to be fast even with a lot of data as input.

All this vagueness to some extent is the fun of the subject

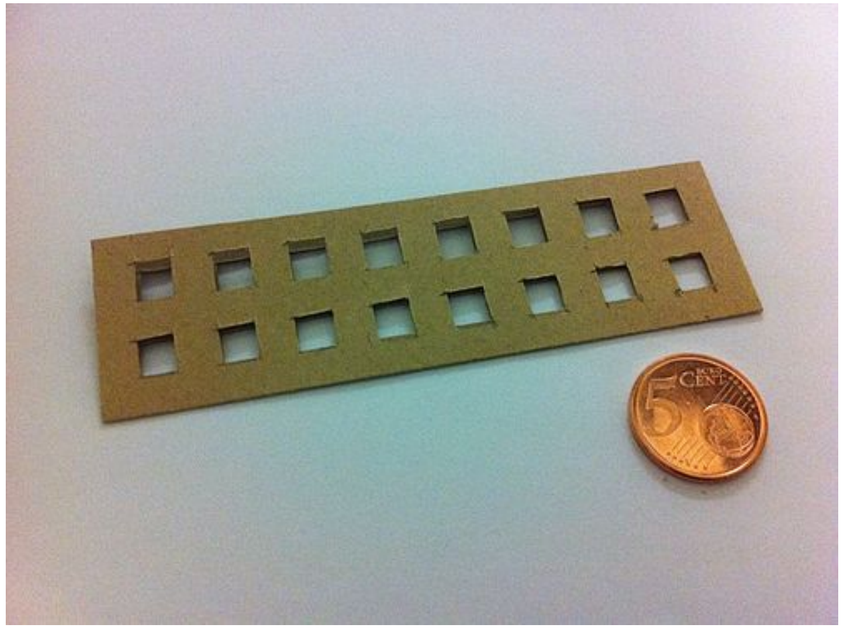
- Algorithm design is both an art and a science.
- Many surprises!
- Many exciting research questions!
- Many real world uses!

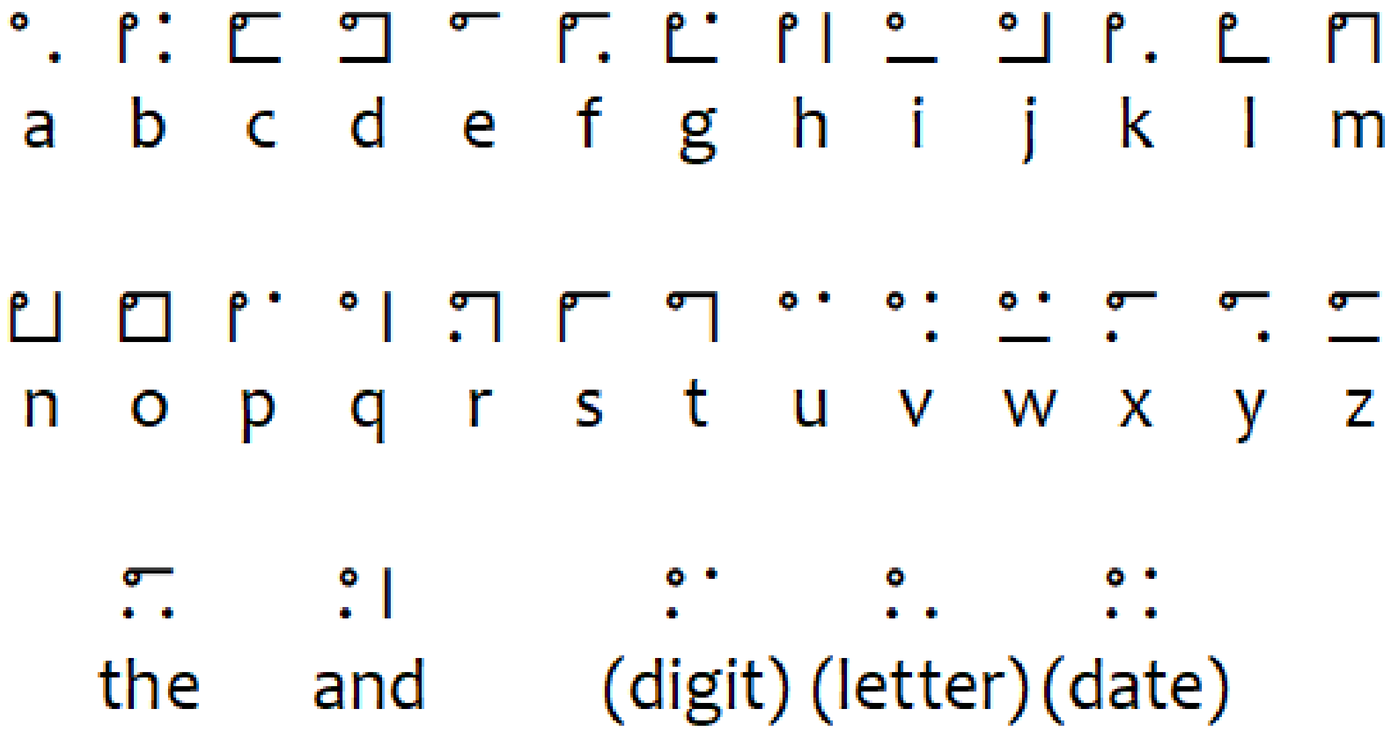
My goals in the tutorials will be

- All of you ace the algorithms course, algorithms portion in the interviews and more essentially, during real world work.
- Some of you take interest in algorithmics and (maybe) consider applying for research type roles
- None of you feel inferior to 'actual' CS people

Back to the story

There was this British combinatorialist Charles Dodgson who lived in and around 1891. Being a math person, he used to get ideas late at night but there were no table lights then and writing our usual cursive English in the dark will lead to stuff that is unreadable in the morning.





Coding Theory

Coding theory is when we encode data not to secure it but to make its transfer more convenient

Dodgeson's Nyctograph was made to make it easier to transfer data from the night to morning easier given the constraints of needing to transcribe it in dark.

Wants to

send N bits

can flip 1 bit

Adds extra K

bits for security

N + K bits

Wants to

receive N bits

A simple and brute-force solution would be adding redundancy. So does **K = N** and copying the bits work?

NO

1 0 1

1 0 1 1 0 1

1 0 0 1 0 1

1 0 1

? ?

1 0 0

1	0	1

1	0	1	1	0	1	1	0	1

1	0	0	1	0	1	1	0	1

1	0	1

But this is too costly!

To send **N** bits, we needed to use **K = 2N** extra bits. Can we do better?

Tip 1: When question is hard, look at simpler question

Wants to send N bits

can flip 1 bit

Adds extra K bits for security

N + K bits

Wants to figure out if there is a flip

Parity Bits

Can anyone give me a **K=1** scheme for this? 

The answer is, counting the number of 1s and making sure it is even.

1	0	1	0

1	0	0	1

1	1	1	1

0	0	0	0

1	1	0	1	1

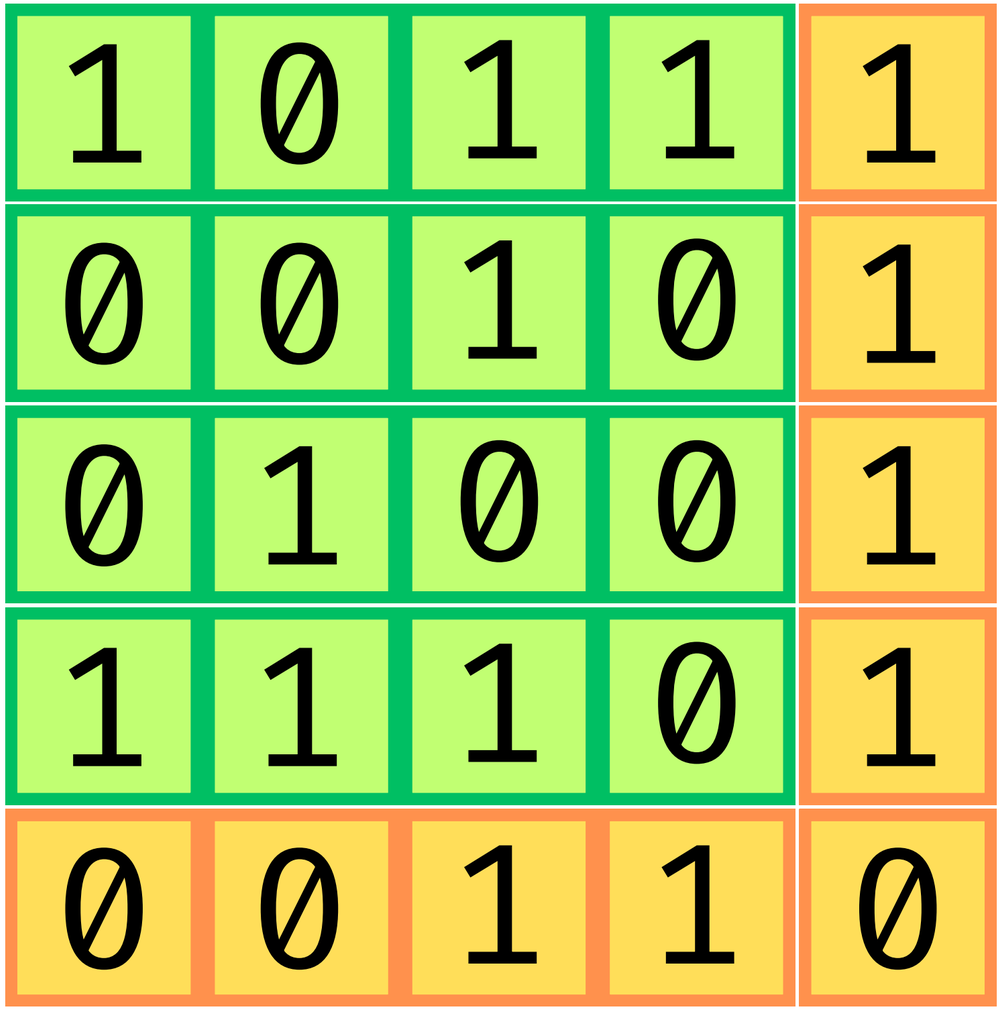
0	0	1	1	0

Can we use this idea to solve our original question? Consider arranging the code in a square

1	0	1	1
0	0	1	0
0	1	0	0
1	1	1	0

Can you give me a way to fix error using **K = 9** ?

The answer is by adding a parity check for every row and column (4 + 4) and then a parity check for these lines (1)



How good is this?

$$
\begin{aligned}
N + K &= \left(\lceil \sqrt{N} \rceil + 1\right)^2 \\
&\leq \left(\sqrt{N} + 2\right)^2 \\
&= N + 4\sqrt{N} + 4 \\
\Rightarrow K &< 4\sqrt{N} + 4
\end{aligned}
$$

Can we do better?

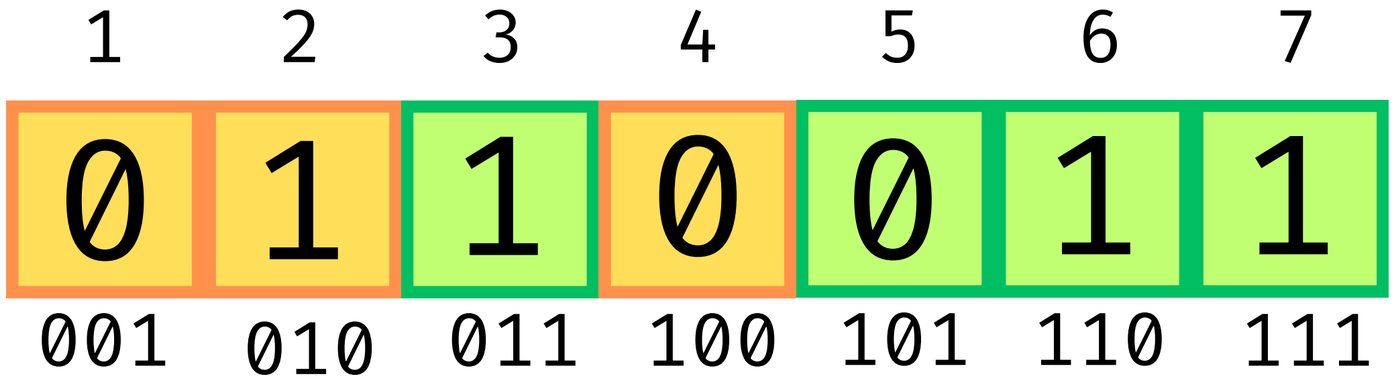
Tip 2: When in doubt, try to check for a mathematical bound

I am gonna just do the math on the board, please refer to notes for the full notation.

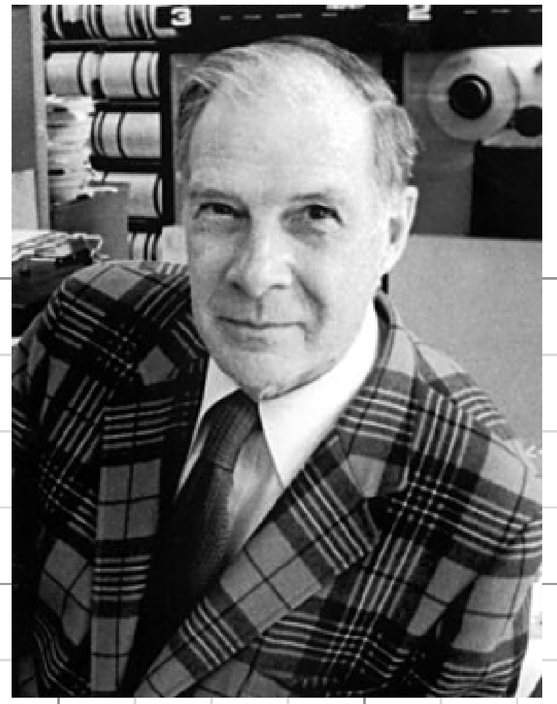
Achieving the Bound

This is just a theoretical bound!

Can it even be achieved? As it turns out yes! And that too by using the parity idea.



For $\mathbf{N = 2^x - x - 1}$, this is optimal and is almost optimal elsewhere.



Other than this being much easier to understand than a lot of the other codes, there is another reason why we might want to use it which I will tell at some other point. (foreshadowing)

These are calling **Hamming Codes**

Reed Solomon Codes

Coming back to our story, IRCTC had copied the QR from the Wikiversity page of something named 'Reed Solomon Codes'.

Now Ria Chopra did try to read that article but didn't understand anything because well Finite Fields, Vandermonde Matrices, Berlekamp–Massey etc. But the key idea doesn't require these fancy concepts.

Wants to send N numbers

can erase S numbers

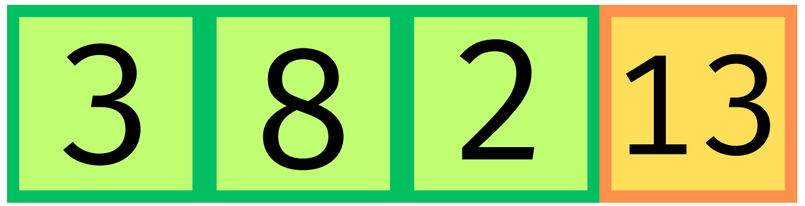
Adds extra K numbers for security

N + K numbers

Wants to receive N numbers

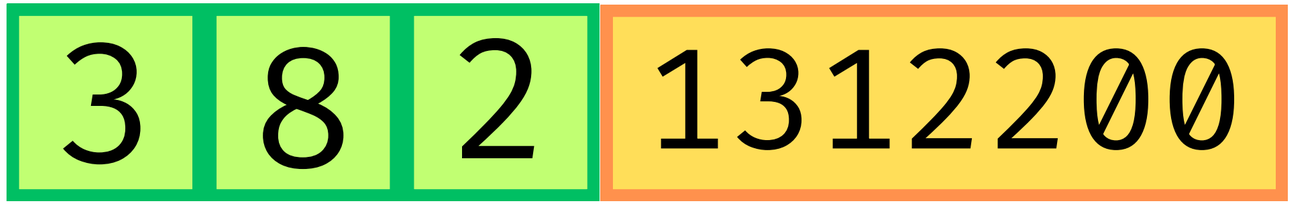
When k=1

A very simple **S=1** solution is adding up all the numbers and attaching the sum at the end.



The same thing can be done with the product...

Or with 2^{m_1} 3^{m_2} 5^{m_3} ...



⋋ = S

One of the solutions on last page can be extended for arbitrary K, can you guess which one?

3	8	2	1312200	1312200	1312200

But this method seems quite slow to encode and decode and leads to huge numbers! For example, (10,10,10) would give **5.9049×10^{14}** . This would quickly overwhelm any storage limits (btw, this is called **Godel Numbering**)

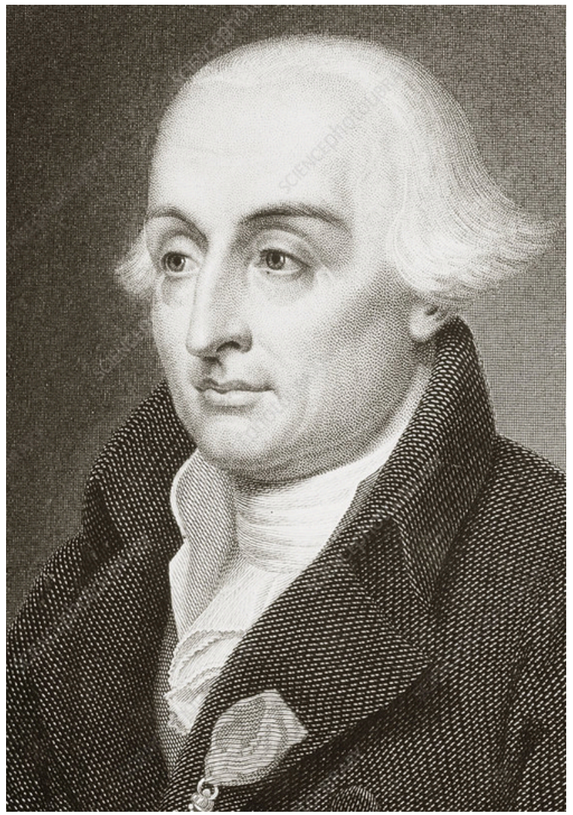
credits to Anshu/Ritika

Polynomials

A seemingly unrelated fact is that **n** points can uniquely define a degree **n-1** polynomial.

That is given **2** points, we can uniquely make a line. Given **3** points, a unique parabola and so on. For the proof of existence and uniqueness refer to the notes.

One way to find such a polynomial is called Lagrange Interpolation,



Idea

Let's try to find a degree 3 polynomial such that $p(1) = 2, p(2) = 5, p(3) = 10, p(4) = 13$.

The idea is to consider $p_1(x) = C_1(x-2)(x-3)(x-4)$ which is 0 for all $2, 3, 4$ and $p_1(1) = p(1)$. We can do so by choosing C_1 to be $-\frac{1}{3}$.

Same way, one can find p_2, p_3, p_4 and add them to get:

$$\begin{aligned} p(x) = &-\frac{1}{3}(x-2)(x-3)(x-4) \\ &+\frac{5}{2}(x-1)(x-3)(x-4) \\ &-5(x-1)(x-2)(x-4) \\ &+\frac{13}{6}(x-1)(x-2)(x-3) \end{aligned}$$

which can be simplyfied to

$$p(x) = -\frac{1}{3}(2x^3 - 15x^2 + 22x - 15)$$

Finally...

Why would I have told you this? Let the message be **m_1, m_2, ..., m_N** then we can consider the **N-1** degree polynomial $\mathbf{p(x) = m\_1 + m\_2\, x + \ldots + m\_N\, x^{(N-1)}}$. Any ideas of how do we complete this?

Simple, Alice evaluate this polynomial at **i = 1 .. N+K** and send **p(1), p(2), ... , p(N+K)**. To find any of the missing values, Bob could just use the remaining at-least **N** points to find the polynomial **p** and hence, recover the original message by observing the coefficients.

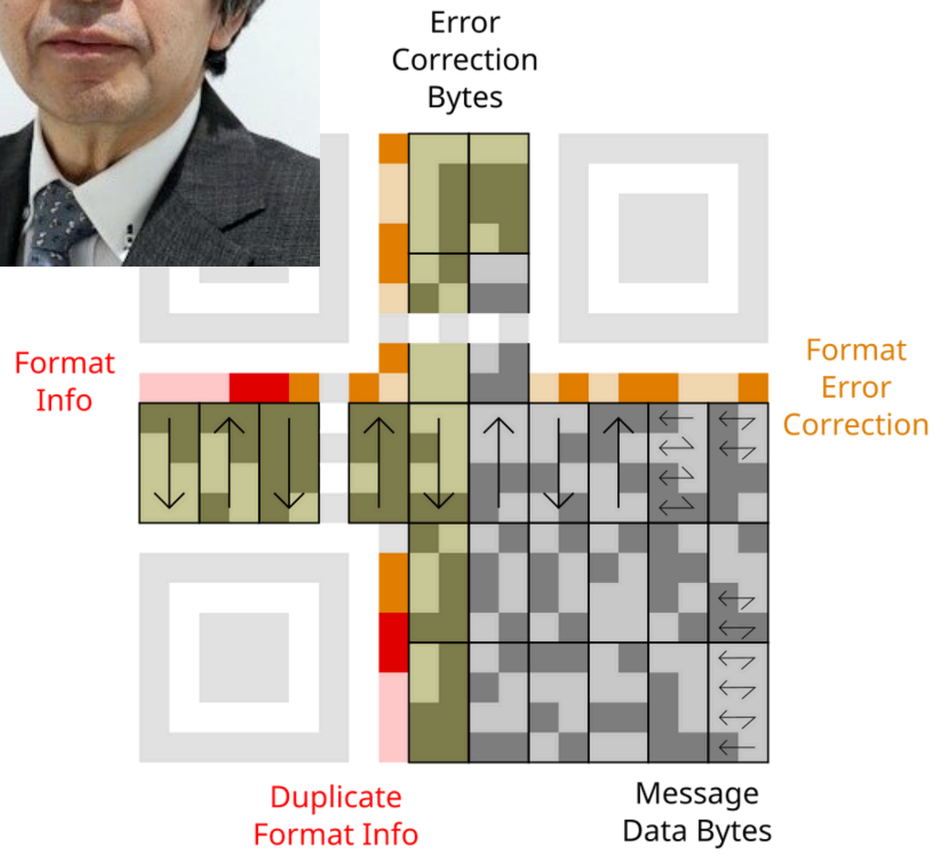
Congratulations, you now understand (a very basic) Reed Solomon code!

QR Codes

QR Codes use the Reed Solomon codes to make sure they are scan-able even with ⅓ of the code are not visible or even outright wrong.

There is a lot of other cool properties of the humble QR code: how we make sure it can be scanned in any orientation (the 3 squares on the corners) and from any angle (the small square in the remaining corner) and the data storage protocols and bit masking for clarity and what not.

QR Codes were invented by Masahiro Hara in 1994 and were inspired by the game Go.



Conclusion

There are a lot of optimisation which one can do in this case. For example, evaluate **0, 1, -1, 2, -2...** to reuse some of the calculations; maybe we interpolate during encoding and evaluate at decoding; what about using $\mathbf{p(x) = m\_1 - m\_2\, x + \ldots + (-1)^{N-1}\, m\_N\, x^{(N-1)}}$ to get smaller values; while on that topic, can we take modulo? Can we handle other types or errors (like changing values)? what about combination of errors? Is this optimal?

As it turns out, a lot of this is possible with Reed Solomon Codes. But to talk about them, we need the language and tools of Finite Field Theory. But this is a CS course and this is the first tutorial.

If you enjoyed this topic, I have put a bunch of links related to this in the notes. **Prof. Amit Kumar Sinhababu** is the resident expert and will be more than willing to teach you the topic further if you are interested. He is taking the course **Algebraic Methods in Theoretical Computer Science (AMTC)** which includes a lot more of this and similar stuff (and the only prereq is Linear Algebra and Algorithms),

But why a Carol poem?

Because Charles Dodgeson aka the mathematician who started this field also wrote a lot of literature under the penname Lewis Carol. So to honor him, the placeholder QR on Wikiversity encoded the lines of one of his most famous poems.

Coding theory deals with correcting errors when the code will be re-permuted, when some bits are more likely to flip than others or when we are okay with messages becoming irrecoverable with some small probability.

Two related topics are compression codes and cryptography. The former encodes in order to make the file smaller while the latter to make sure only the designated receiver can read a message (foreshadowing)

Next Time...

- Big Oh Notation
- Ad-Hoc Algorithms
- Introduction to Graphs