Algorithm Design Paradigms

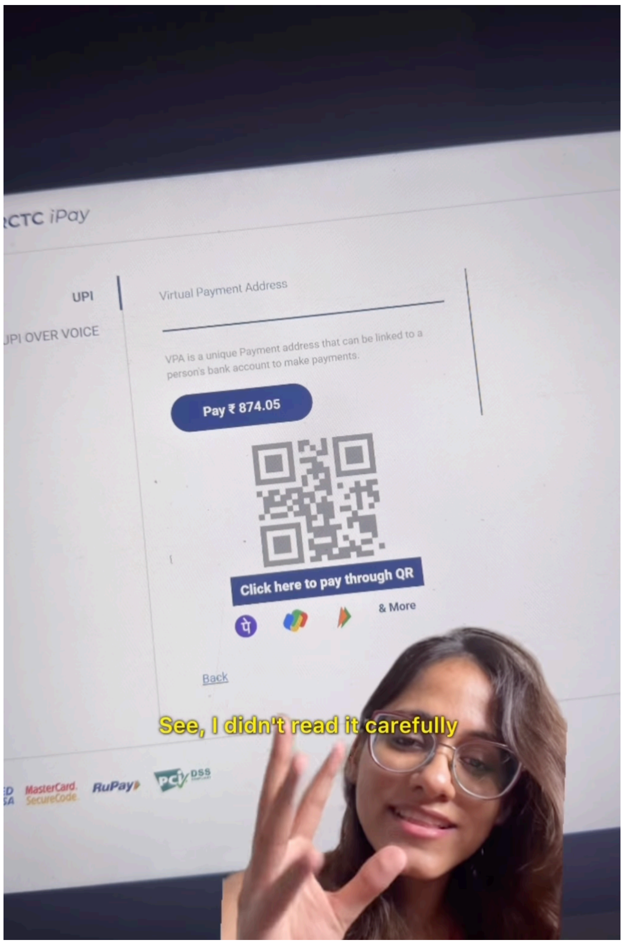
Handout I

10 January 2026

§1 ♠ 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**“.


CTC iPay
UPI
UPI OVER VOICE
Virtual Payment Address
VPA is a unique Payment address that can be linked to a person's bank account to make payments.
Pay ₹ 874.05
Click here to pay through QR
& More
Back
See, I didn't read it carefully
MasterCard SecureCode
RuPay
PCI DSS

If anyone knows the meaning of this phrase, this is your opportunity to get a chocolate. Otherwise, I will tell you the origin in a moment.

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.

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?

While the exact mechanisms of how QR codes work is beyond this course and definitely this tutorial, feel free to ask Prof. Amit Kumar Sinhababu about it. He is the resident expert in this subfield of algorithms called Coding Theory.

§2 Welcome to the Course

Hello, I am **Arjun Maneesh Agarwal**. I am in BSc II and I along with **Harshitha Mucherla** (BSc II) and **Vardhan Kumar Ray** bhaiya (MSc CS I) am a TA for this course.

My goal with this course and the tutorials is to make sure
- All of you ace the algorithms part of your 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

I will routinely cover some additional material which will not be for the exam but I feel are nice to know. This material will be marked with ♠.

While I personally do all things algo, my interest broadly lies in using algorithmic techniques to model, predict and understand social and behavioral 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.

If you have any doubts, feel free to interject. It is rude but we don’t mind.

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.

§3 What is an Algorithm?

Before we get to designing and analyzing algorithms, let’s pause and briefly question what ‘algorithm’ actually means. To quote Hannah Fry,

> *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.
>
> *An algorithm is simply a series of logical instructions that show, from start to finish, how to accomplish a task. By this broad definition, a cake recipe counts as an algorithm. So does a list of directions you might give to a lost stranger. IKEA manuals, YouTube troubleshooting videos,*

even self-help books – in theory, any self-contained list of instructions for achieving a specific, defined objective could be described as an algorithm. But that's not quite how the term is used. Usually, algorithms refer to something a little more specific. They still boil down to a list of step-by-step instructions, but these algorithms are almost always mathematical objects. They take a sequence of mathematical operations – using equations, arithmetic, algebra, calculus, logic and probability – and translate them into computer code. They are fed with data from the real world, given an objective and set to work crunching through the calculations to achieve their aim. They are what makes computer science an actual science, and in the process have fuelled many of the most miraculous modern achievements made by machines.

— Hannah Fry

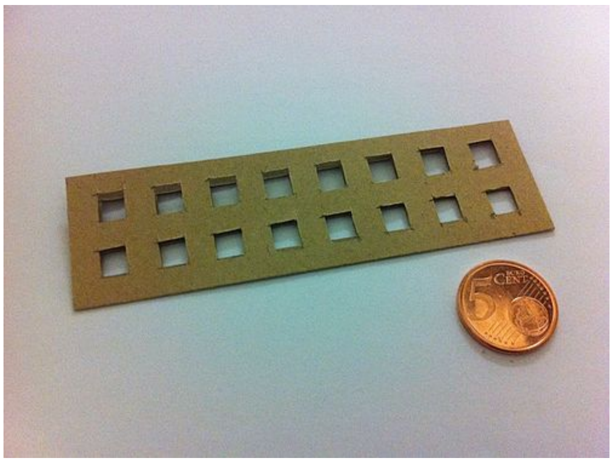
We will take Prof. Fry's definition as the gospel as trying to go into more details will open up questions which are more philosophical than we wish to be here. As a final remark, algorithmics is both an art and a science and hence leads to a lot of surprises and approachable open problems.

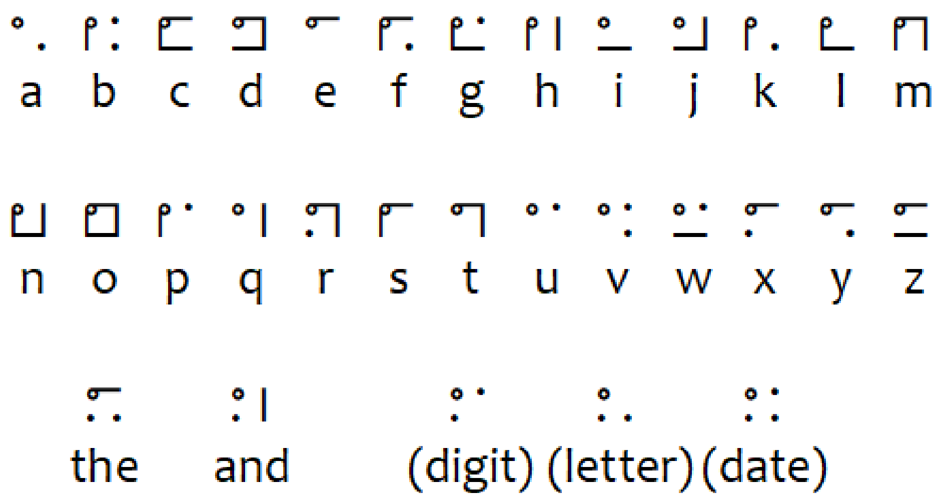
Back to the story!

§4 ♠ Problems of Communication

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.

So what did he do? He made a new script using dot's and squares which could be written neatly even in the dark of the night.





This was perhaps the birth of coding theory. Coding theory is when we encode data not to secure it but to make its transfer more convenient (in this case transfer from night to morning).

Let's take a more concrete example

Exercise

Alice is given N bits, she should attach K bits on the end and send to Bob
- At most one bit of the $(N + K)$ bits might be flipped
- Bob needs to recover the original N bits.

We want to minimize K.

A simple and brute-force solution would be adding redundancy. So does $K = N$ and copying the bits work? **NO!** Consider for $N = 3$,

$$101 \xrightarrow{\text{encode}} 10110\mathbf{1} \xrightarrow{\text{flip 6th bit}} 10110\mathbf{0}$$
$$100 \xrightarrow{\text{encode}} 10\mathbf{0}100 \xrightarrow{\text{flip 3rd bit}} 10\mathbf{1}100$$

and we have an issue with decoding!

So what can we do? Use this idea but make 2 extra copies.

$$101 \xrightarrow{\text{encode}} 101101101$$
$$100 \xrightarrow{\text{encode}} 100100100$$

But this is way too expensive! Can we do better? Let's look at a simpler question.

Exercise

Bob only needs to tell if a bit has been flipped and atmost 1 bit has been flipped.

Can anyone give me a $K = 1$ scheme for this? I'll give you chocolate.

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

$$\begin{aligned} 101 &\to 101\mathbf{0} \\ 100 &\to 100\mathbf{1} \\ 000 &\to 000\mathbf{0} \\ 111 &\to 111\mathbf{1} \\ 1111 &\to 1111\mathbf{0} \end{aligned}$$

So if any bit flips, the parity of number of 1s changes and Bob finds out. Can we use this scheme to solve our original question? Consider arranging a 16 bit code (1011001011000100) in a 4×4 square.

$$\begin{matrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

Can you give me a way to fix error using $K = 9$? There is a chocolate for this!

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

$$\begin{matrix} 1 & 0 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 1 & 0 & \mathbf{1} \\ 1 & 1 & 0 & 0 & \mathbf{0} \\ 0 & 1 & 0 & 0 & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{matrix}$$

And now in case of an error, we will get a parity error in a row and column and tell us the exact location of the error. This gives:

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

Is this the best we can do?

Let's try to mathematically bound K.

We want to define two functions,

$$\begin{aligned} \text{encode} &: \text{messages of length } N \to \text{messages of length } N + K \\ \text{decode} &: \text{messages of length } N + K \to \text{messages of length } N \end{aligned}$$

such that decode $\circ$ flip_i $\circ$ encode $=$ id for all $i = 0..N + K$ (where $\text{flip}_0 = \text{id}$).

As a great man said, "Math just provides a language to talk about problems more easily." From this point, we can just follow our nose:

Encode clearly maps the 2^N messages to a subset of the 2^{N+K} messages.

We can model the constraint as: if Bob receives c then it can be:

- c itself (no error), or
- c with the i-th bit flipped, for some $i = 1...N + K$

So each codeword c has to be identified by $N + K + 1$ possible received words.

This implies $2^N * (N + K + 1) \leq 2^{N+K} \Rightarrow N + K + 1 \leq 2^K$.

But this is a theoretical bound. Can we even achieve it? As it turns out, YES! and that too by using the parity idea!

Instead of trying to identify the position of error using the position, could we use something else? We studied it today.

The answer is binary encoding! Look at the binary expansion of $1..N$ and do the parity check on bits where the i^{th} digit in binary expansion is 1.

Clearly, $1 \leq i \leq \lceil \log_2(n) \rceil$. We add one final bit to check if the error is in the error correcting bits (in that case we just read the message!).

Finally, notice that instead of storing these error correcting bits at the start or the end, we could store them at their natural positions $1_2, 10_2, 100_2, 1000_2, \ldots$ aka the powers of 2! This also makes the case for appending the parity bit at the start (position 0).

This gives us optimality when $N = 2^x - x - 1$ 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. These are calling **Hamming Codes**

§5 ♠ Reed Solomon Codes

Coming back to our story, IRCTC had copied the QR from the Wikipedia 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.

Exercise

Alice is given N integers, she should attach K integers on the end of the message and send to Bob.
- At most K numbers of the $(N + K)$ numbers might be erased.
- Bob needs to recover the original N numbers. Bob knows the position of the numbers that were erased but not their values.

This is much a more interesting question. What can we do?

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.

One way of doing this is called the 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

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

Remark

If you know linear algebra, something here must feel like it could be done with vectors and matrices. In that case, you are correct and that is how the Vandermonde matrix appears in this case.

Theorem 5.1. *Given $(x_1, y_1), ..., (x_n, y_n)$, there is a unique degree $n-1$ polynomial p such that $p(x_i) = y_1$*

Proof. For the sake of contradiction, let there be two distinct degree $n-1$ polynomials f, g such that $\forall i \in [n], f(x_i) = g(x_i) = y_i$.

This implies $h(x) = f(x) - g(x)$ is a polynomial of degree atmost $n-1$ and has roots $x_1, x_2, ..., x_n$. Thus, by factor theorem, $h(x) = C(x-x_1)(x-x_2)...(x-x_n)$. But this is a contradiction as this implies the degree of $h(x)$ is atleast n (as $h(x) = 0$ as f and g are distinct).

Thus, our initial assumption must be false and one can't find two distinct polynomials f, g such that $\forall i \in [n], f(x_i) = g(x_i) = y_i$.

We can construct f using Lagrange Interpolation to show existence. □

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 $p(x) = m_1 + m_2 x + ... + m_N x^{N-1}$. 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!

It can also be shown that this is optimal.

As it further turns out, we can also fix errors which are a combination of erasures and value changes. More so, we don't need the full power of $\mathbb{Z}$ and instead can work with smaller subsets (usually numbers uptill some 2^m). This is where Finite Fields come in. Again, all of it is very interesting but we unfortunately don't want this to turn into a math course especially a field theory course.

§6 Conclusion

Coding theory also 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 probability. The key question in any scheme tend to be: number of extra bits, time to encode and time to decode.

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. We shall see both of them later in this course.

While there is a lot of nerding out I could do about the humble QR code. About 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; I think this is more than enough for the first class.

So why would the Reed-Solomon codes's wikiversity have this? Because Charles Dodgeson aka the mathematician who started this field also wrote a lot of literature under the penname Lewis Carrol. So to honor him, the example QR used encoded the lines of one of his most famous poems.

If you enjoyed this topic, I have put a bunch of links related to this on the tutorial website. As mentioned before, 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 would probably include a lot more of this and similar stuff.

§7 Extra Links

If you don't watch anything else from here, consider watching Prof. Fry's talk (first link). It is aimed at the general public but covers a lot of concepts you should probably start thinking about.

- Hannah Fry (goat!) on Algorithms
- Dylan Bettie on Algorithms
- 3 Blue 1 Brown on Hamming codes
- vcubingx on Reed Solomon codes (perhaps the most intuitive)
- A rather similar treatment of Reed Solomon as yours truly.
- Mary Wooter's course on Coding Theory (I haven't watched it but she is a famously good professor)
- A very rigorous explanation of Reed Solomon codes
- Wikiversity for Reed Solomon Codes, aimed at programmers