# CS250/EE387 - LECTURE 16 - RS codes as Regenerating codes

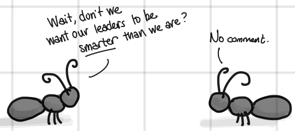
AGENDA.

0. Application of locality: distributed storage
1. What's the model?
2. RS codes are a bad idea for distributed storage
3. RS codes are a great idea for distributed storage!
4. COURSE RECAP

TODAY'S ANT FACT

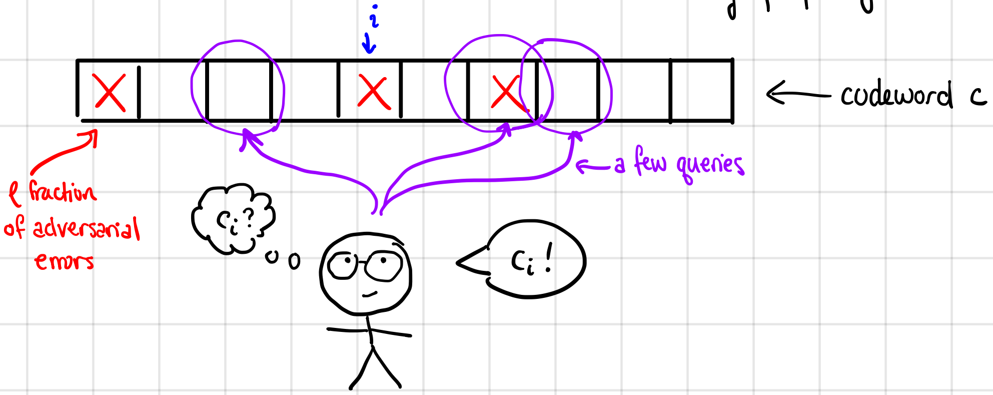
Indian Jumping Ants can shrink and re-grow their brains. When a queen dies, the female workers fight to see who will be the new queen. The winner undergoes physical changes, including shrinking her brain! But if she loses her role, she can undo these changes!

Oh no, our final projects are due next week! Time to regrow those brain cells!



## 0. APPLICATION (?) of LOCALITY

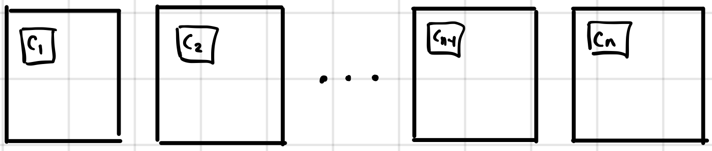
So far, we've seen LCC's, which have the following property:



This seems like it should come in handy in the following DISTRIBUTED STORAGE setting:

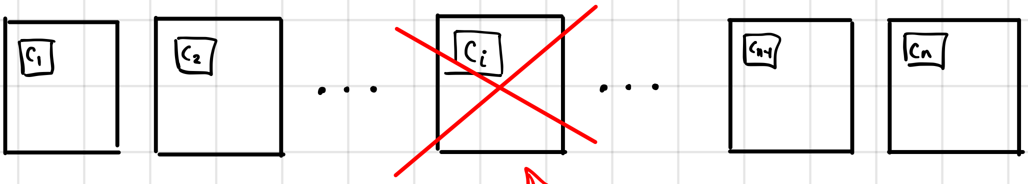
$$\boxed{f} = (x_1, \ldots, x_k) \in \mathbb{F}_q^k \xrightarrow{\text{encode with an LCC}} (c_1, c_2, \ldots, c_n) \in \mathbb{F}^n$$

↓ STORE on n different nodes



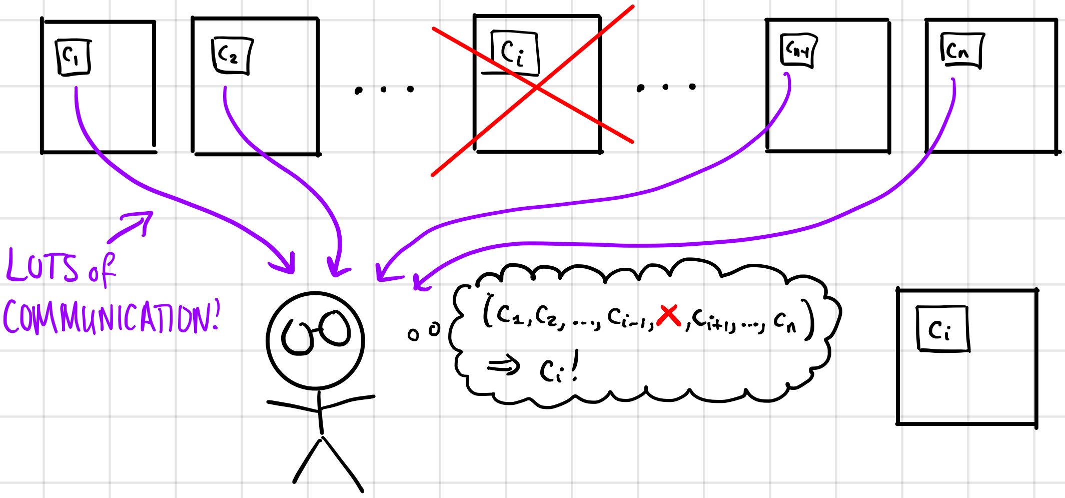
(each node also holds some other stuff, say encodings of other files in the system... but let's just focus on one file.)

Now suppose that a server fails, and I'd like to repair it to maintain the system.

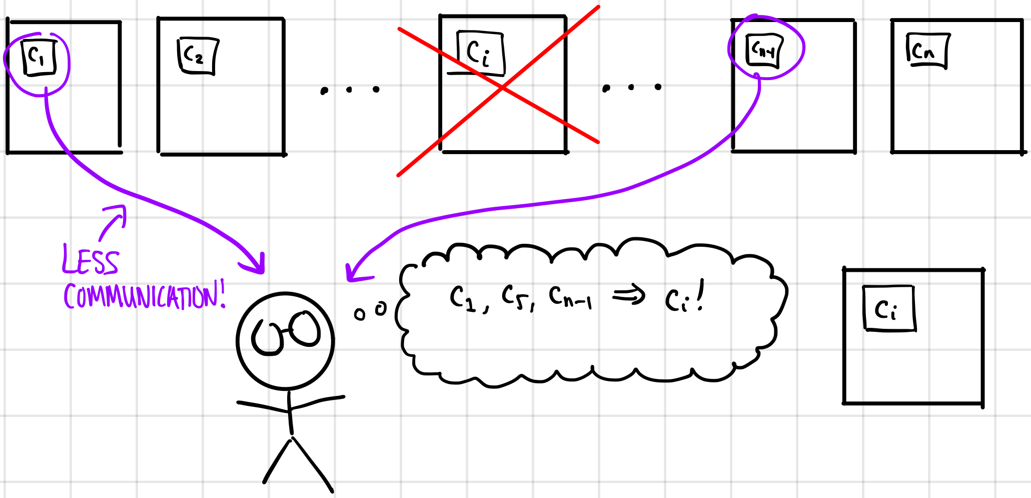


This guy is down.
(Say for long enough that we don't want to wait for it to come back up).

OPTION 1: Download all the surviving blocks and correct the error.



OPTION 2: LOCALLY CORRECT the error, and download just the blocks you need to do that.



It turns out that communication is EXPENSIVE (and is a bottleneck in distributed storage systems) so this is a win.

① What's the model here?

LOCALITY seems useful. But are LCCs the right tool for the job?

ANSWER: Not really.

(a) The right model is ERASURES, not ERRORS.

(b) 98% of the time*, only ONE server is down.

* Based on a study of the Facebook Warehouse cluster.

Instead what do we want?

(1) Best trade-off between RATE and DISTANCE possible - aka an $\underbrace{\text{MDS code}}_{\text{Recall this means } n-k+1=d}$.

- We want to handle as many failures as possible in the worst case.

(2) Every symbol can be obtained from not-too-many other symbols.

- When there is only 1 failure, we'd like to repair it with minimal communication.

② RS CODES are a BAD IDEA for DISTRIBUTED STORAGE.

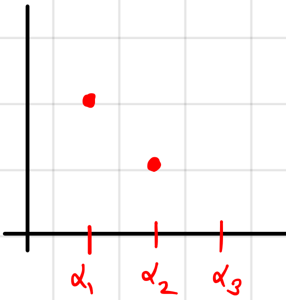
(1) MDS Code ✓

(2) Every symbol can be obtained by not-too many other symbols ✗

(2) doesn't hold:

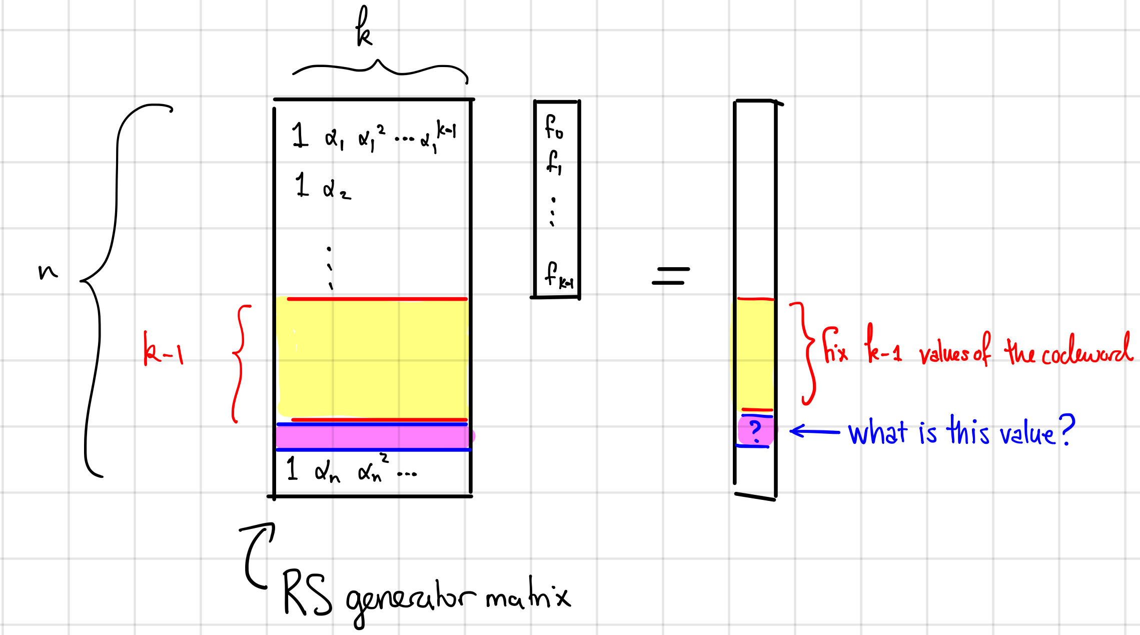
- Suppose $f \in \mathbb{F}_q[X]$, $\deg(f) < k$.
- I <u>NEED</u> $k$ evaluation pts $f(\alpha_1), \ldots, f(\alpha_k)$ to say ANYTHING at all about $f(\alpha_{k+1})$

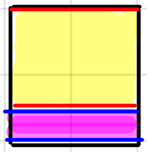
eg, suppose $f(X)$ is a quadratic and goes through these 2 points:
what is $f(\alpha_3)$? COULD BE ANYTHING.



**CLAIM:** Given any $k-1$ symbols of an RS codeword $c$, a $k^{th}$ symbol could be anything in $\mathbb{F}_q$.

Proof by picture:



Now this $k \times k$ matrix  is full rank, so for ANY value of $\boxed{?}$, there is some $(f_0, \ldots, f_{k-1})$ that is consistent. So $\boxed{?}$ could be anything.

What does this mean for RS codes?

We need to query $k$ symbols just to get one – but $k$ is enough to recover the whole message!

So that's really wasteful.

So can we find some other code satisfying (1) and (2)?

NO! Actually that argument works for any MDS code, not just RS codes. So:

If (1) MDS Code ✓

then (2) Every symbol can be obtained by not-too-many other symbols ✗

Two WAYS around this:

WAY 1: Give up on MDS. ← This is really interesting and the buzzword is "Locally Recoverable Code." We won't talk about it.

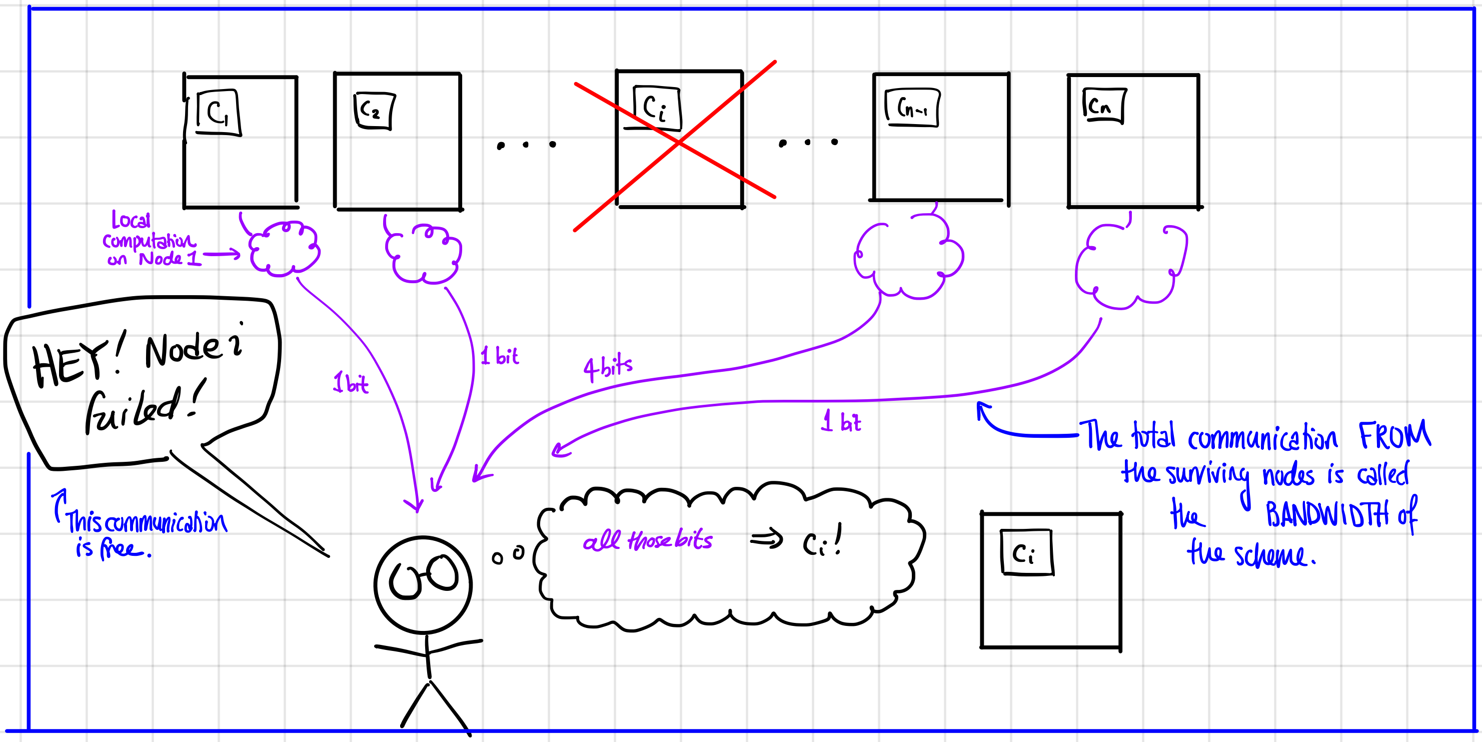
WAY 2: Rephrase (2) ← We will talk about this.

We will instead shoot for:

(1) MDS Code

(2) Every symbol can be obtained by not-too-many BITS from other symbols.

In pictures the model is this:



This communication is free.

The total communication FROM the surviving nodes is called the BANDWIDTH of the scheme.

Such a code is called a REGENERATING CODE.
There's tons of super cool work on these that I won't talk about.
But for today ...

"THM." Reed-Solomon Codes ARE good regenerating codes.

## ③ RS CODES are a GREAT IDEA for distributed storage!

For simplicity let's focus on $k = n/2$, $n = q$, $q = 2^t$.
So a codeword of $RS_q(\mathbb{F}_q, q, q/2)$ looks like:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $f(0)$ | $f(\gamma)$ | $f(\gamma^2)$ | | | | | | | | | | $f(\gamma^{q-1})$ |

for a primitive elt $\gamma$.

Say $f(0)$ fails. ← Works with any node, but for concreteness say it's $f(0)$.

**CLAIM** (which we will show)

It is possible to download ONE BIT from $f(\gamma^i)$ for $i = 1, \ldots, q-1$, and recover $f(0)$.

Notice this is $q-1$ BITS total, while the naive scheme would download $k = q/2$ whole symbols, each are $\lg(q)$ bits — so that's $\frac{q \lg(q)}{2}$.

So the CLAIM is BETTER than the naive scheme!

**CLAIM** (which we will not show)

This is optimal*

*For a linear scheme, for an MDS code.

To prove the first CLAIM, we will need the following algebra facts:

**FACT:** $\mathbb{F}_{2^t}$ is a vector space over $\mathbb{F}_2$.

So we can think of $\alpha \in \mathbb{F}_{2^t}$ as a vector $\vec{\alpha} \in \mathbb{F}_2^t$ if we want.
(of course, this is for the additive structure only).

**FACT.** Let $P(X) = X + X^2 + X^4 + \cdots + X^{2^{t-1}}$. Then

(a) $P : \mathbb{F}_{2^t} \to \mathbb{F}_2$ is $\mathbb{F}_2$-linear.

aka, $P(\alpha) \in \mathbb{F}_2 \; \forall \, \alpha \in \mathbb{F}_{2^t}$, and $P(\alpha+\beta) = P(\alpha) + P(\beta)$

(b) All $\mathbb{F}_2$-linear fns $\psi : \mathbb{F}_{2^t} \to \mathbb{F}_2$ have the form $\psi(X) = P(\gamma \cdot X)$ for some $\gamma \in \mathbb{F}_{2^t}$.

(c) "Morally" we should think of $P(\alpha \cdot \beta)$ as $\langle \vec{\alpha}, \vec{\beta} \rangle$ for $\vec{\alpha}, \vec{\beta} \in \mathbb{F}_2^t$.

In fact, there always exists a basis so that if $\vec{\alpha}$ is $\alpha$ written out w/r/t this basis, then

$$P(\alpha \cdot \beta) = \langle \vec{\alpha}, \vec{\beta} \rangle = \sum_{i=1}^{t} \alpha_i \beta_i$$

($P(X)$ is usually called the "field trace.")

"pf" of FACT (a):

To see $P(\alpha+\beta) = P(\alpha) + P(\beta)$, recall $(\alpha+\beta)^2 = \alpha^2 + \beta^2$ in $\mathbb{F}_{2^t}$.

To see $P(X) \in \mathbb{F}_2$, notice $P(X)^2 = P(X)$, which is only true for 0 and 1.

Now that we have these facts, we can prove the CLAIM. Recall $q = 2^t$.

By RS duality, $RS_q(\mathbb{F}_q, q, \frac{q}{2})^{\perp} = RS_q(\mathbb{F}_q, q, \frac{q}{2})$

So for all $f, g \in \mathbb{F}_q[X]$ w/ degree $< k = q/2$,

$$0 = \sum_{\alpha \in \mathbb{F}_q} f(\alpha) \cdot g(\alpha)$$

$$f(0) \cdot g(0) = \sum_{\alpha \in \mathbb{F}_q \setminus \{0\}} f(\alpha) \cdot g(\alpha)$$

For any $\gamma \in \mathbb{F}_{2^t}$, let $g_\gamma(X) = \dfrac{P(\gamma \cdot X)}{X} = \gamma + X + X^3 + X^7 + \cdots + X^{2^{t-1}-1}$.

Then $\deg(g_\gamma) = 2^{t-1} - 1 = \frac{q}{2} - 1 = k - 1$.

So we may plug in $g_\gamma$ for $g$ above:

$\forall f \in \mathbb{F}_q[X]$ s.t. $\deg(f) < q/2$ :

$$f(0) \cdot g_\gamma(0) = \sum'_{\alpha \in \mathbb{F}_q \setminus \{0\}} f(\alpha) \cdot g_\gamma(\alpha)$$

Plug in $g_\gamma$ for $g$

$$f(0) \cdot \gamma = \sum_{\alpha \in \mathbb{F}_q \setminus \{0\}} f(\alpha) \cdot \frac{P(\gamma\alpha)}{\alpha}$$

Def of $g_\gamma$

$$P(f(0) \cdot \gamma) = P\left( \sum_{\alpha \in \mathbb{F}_q \setminus \{0\}} f(\alpha) \cdot \frac{P(\gamma\alpha)}{\alpha} \right)$$

Take $P(\cdot)$ on both sides

$$P(f(0) \cdot \gamma) = \sum_{\alpha \in \mathbb{F}_q \setminus 0} P\left( f(\alpha) \cdot \frac{P(\gamma\alpha)}{\alpha} \right)$$

$P(\cdot)$ is $\mathbb{F}_2$-linear

$$\langle \vec{f(0)}, \vec{\gamma} \rangle = \sum_{\alpha \in \mathbb{F}_q \setminus 0} \left\langle \vec{f(\alpha)}, \overrightarrow{\frac{P(\gamma\alpha)}{\alpha}} \right\rangle$$

$P(\alpha \cdot \beta) \approx \langle \vec{\alpha}, \vec{\beta} \rangle$, morally

$$\langle \vec{f(0)}, \vec{\gamma} \rangle = \sum_{\alpha \in \mathbb{F}_q \setminus 0} P(\gamma\alpha) \langle \vec{f(\alpha)}, \vec{\alpha^{-1}} \rangle$$

$P(\gamma\alpha) \in \mathbb{F}_2$, so it's just a scalar.

So for all $\vec{\gamma} \in \mathbb{F}_2^t$, we have

$$\langle \vec{f(0)}, \vec{\gamma} \rangle = \sum_{\alpha \in \mathbb{F}_q \setminus 0} P(\gamma\alpha) \langle \vec{f(\alpha)}, \vec{\alpha^{-1}} \rangle$$

Recall the goal is to find f(0). So the algorithm is:

This is one bit per node, as promised!

**ALG.** (Assuming f(0) has failed).

- The node holding f(α) returns $b_\alpha = \langle \vec{f(\alpha)}, \vec{(\alpha^{-1})} \rangle \in \mathbb{F}_2$
- We compute $\langle \vec{e_i}, \vec{f(0)} \rangle = \sum_{\alpha \in \mathbb{F}_q \setminus 0} P(\beta_i \cdot \alpha) \cdot b_\alpha \in \mathbb{F}_2$ for all $i$, where $\beta_i \in \mathbb{F}_{2^t}$ s.t. $\vec{\beta_i} = \vec{e_i} \in \mathbb{F}_2^t$
- Let $f(0) = \left( \langle \vec{e_1}, \vec{f(0)} \rangle, \langle \vec{e_2}, \vec{f(0)} \rangle, \ldots, \langle \vec{e_t}, \vec{f(0)} \rangle \right)$

That's it! This feels a bit magical, but actually it generalizes to some other parameter regimes and also turns out to be optimal!

See [Guruswami, W. '16], [Dau, Milenkovic '17], [Tamo-Ye-Barg '17] for more.

**The point:**

- For distributed storage, a different notion of locality is appropriate. This is good news since even though RS codes are NOT good LCCs, they ARE good regenerating codes!
- Also, this is kind of a neat fact about polynomial interpolation.

## ④ Course Recap

This is the last lecture... WHAT HAVE WE LEARNED?

### WHAT HAVE WE LEARNED?

- Fundamental trade-offs between RATE and DISTANCE
  - The "correct" trade-off for binary codes is still open, but over large alphabets it is attained by...
- REED-SOLOMON CODES and "LOW-DEGREE POLYS DON'T HAVE TOO MANY ROOTS."
  - OMG the BEST code!
- How to decode RS codes, and how to use this to get efficiently decodable binary codes.
  - Reed-Muller, BCH, concatenation, oh my!
- Brief detour into RANDOM ERRORS – and we can get the same trade-offs with LIST-DECODING!
  - Capacity = $1 - H_q(p)$ either way!
- We can do list-decoding (also list-recovery) EFFICIENTLY w/ the GURUSWAMI-SUDAN Algorithm! And we can modify this to achieve capacity by FOLDING.
  - STEP 1: INTERPOLATE. STEP 2: ROOT-FIND. STEP 3: PROFIT.
- We talked about RM codes and locality!
  - Plus, local-list-decoding, and just now regenerating codes!
- Along the way, APPLICATIONS!
  - Crypto, Compressed Sensing, Group testing, Heavy Hitters, Learning theory, Storage, (communication, QR codes, hat puzzles, ...)

## THE MORAL(s) of the STORY :

(1) Low-degree polynomials don't have too many roots.
and this fact is unreasonably useful !

(2) Error correcting codes show up all over the place.
maybe even in your own research!

# QUESTION TO PONDER

What can error correcting codes do for you?