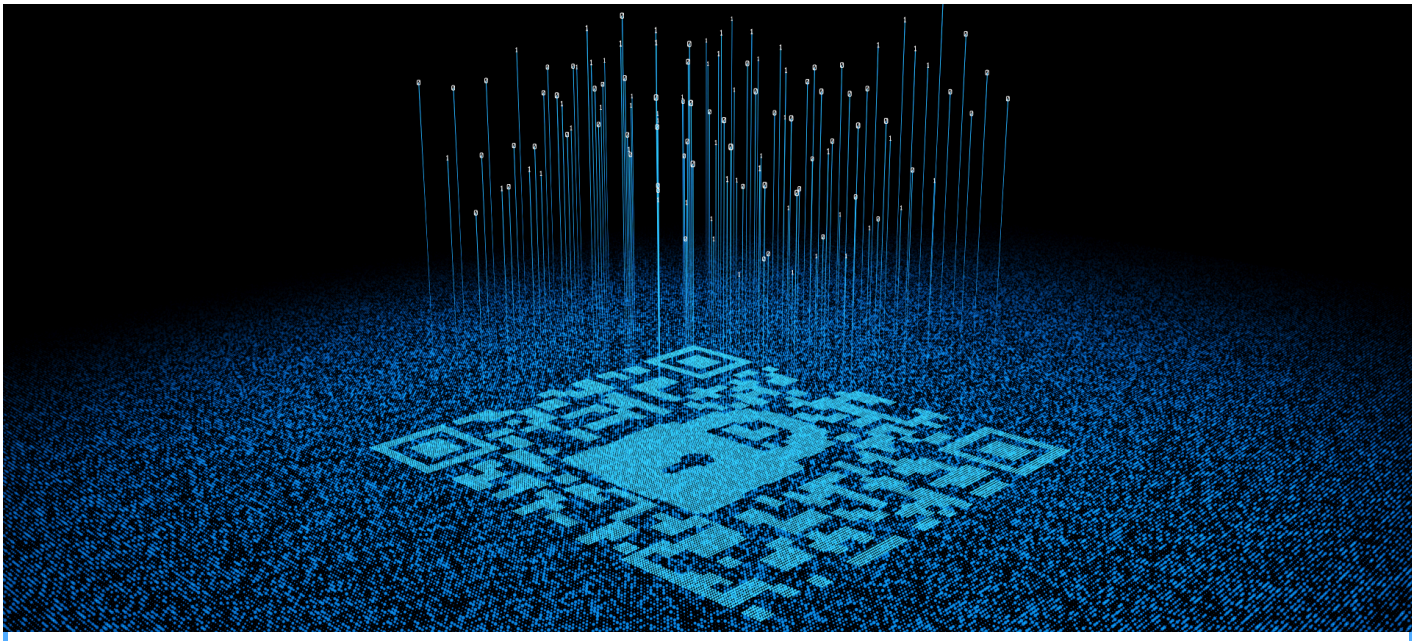
QR Code e Correzione degli Errori: Come Funzionano i Codici Reed-Solomon

DI MASSIMO VAILATI

10/09/2026

ALGORITMI E PROGRAMMAZIONE

LICEO, TECNICO



I **QR code** (da *Quick Response code*) sono un tipo di **codice a barre bidimensionale** inventato nel 1994 dalla società giapponese Denso Wave. A differenza dei codici a barre tradizionali (quelli con le linee verticali), il QR code usa **quadrati neri e bianchi disposti su una griglia** e può contenere molte più informazioni, essere letto da qualsiasi angolazione, resistere a graffi e danni grazie alla **correzione degli errori** e poter memorizzare URL, testi, contatti o pagamenti.
Rispetto ai codici a barre hanno però dimensioni maggiori, necessitano di scanner o smartphone moderni, la stampa è più complessa e la lettura meno veloce in applicazioni ad alto volume (come le casse dei supermercati o i magazzini).

COME FUNZIONA UN QR CODE

Un QR code codifica informazioni trasformandole in **bit**, cioè sequenze di 0 e 1.
Il processo è questo:

1. **Codifica dei dati**
Il contenuto (es. un link) viene convertito in codice binario.

2. **Organizzazione nella griglia**
I bit vengono distribuiti nei piccoli quadratini bianchi e neri.

3. **Elementi di orientamento**
Tre grandi quadrati agli angoli aiutano lo scanner a capire:

- dove si trova il codice
- come è orientato
- la dimensione della griglia

4. **Lettura con una fotocamera**
Quando lo scanner (per esempio la fotocamera dello smartphone) legge il QR code:

- riconosce i quadrati di riferimento
- ricostruisce la griglia
- traduce i quadratini in dati digitali
- mostra il contenuto (per esempio apre un sito).

PERCHÉ FUNZIONANO ANCHE SE ROVINATI?

Una delle caratteristiche più interessanti è la **correzione degli errori**.
I QR code usano un sistema matematico chiamato **codice di correzione** (in particolare basato sull'algoritmo Reed-Solomon error correction).
Questo significa che:

- una parte dei dati è **ridondante**
- il sistema può **ricostruire le parti mancanti**

A seconda del livello di correzione scelto, il QR code può essere letto anche se **fino al 30% dell'immagine è danneggiata**.
È per questo che:

- funzionano anche se sono graffiati
- si leggono anche se un po' sporchi
- a volte funzionano anche se coperti parzialmente.

GENERARE QR CODE

Generare un **QR code** oggi è molto semplice e ci sono diversi modi, sia online che con programmi o linguaggi di programmazione. Un sito che offre gratuitamente un servizio di base è ad esempio **https://www.qr-code-generator.com/**.

ATTIVITÀ PRATICA N.1

Quanto puoi rovinare un QR code prima che smetta di funzionare?

Obiettivo
Dimostrare che i QR code contengono **ridondanza e sistemi di correzione degli errori**. È un **esperimento molto semplice e sorprendente** da fare in classe per mostrare perché i QR code funzionano anche se sono rovinati. Gli studenti lo capiscono subito perché lo vedono succedere davvero. È possibile utilizzare un qualsiasi QR code esistente o generato dagli studenti.

1. Test iniziale
Gli studenti scansionano il QR code. Funziona normalmente.

2. Primo danneggiamento
Coprire **una piccola parte** del QR code (usare un pennarello se stampato o modificare l'immagine). Qui il QRCode di hubscuola.it in cui sono state alterate due parti una in nero e una in bianco.

Chiedi alla classe: “Secondo voi funziona ancora?”
Gli studenti lo scannerizzano. Di solito funziona ancora.

3. Danneggiamento maggiore
Colora una zona del codice oppure copri **10-20% della superficie**. Qui il QRCode di hubscuola.it in cui, oltre al danno precedente, sono stati colorati dei punti e aggiunto un ulteriore deterioramento in rosso.

Scansione di nuovo. Spesso funziona ancora.

4. Sfida finale
Chiedi agli studenti di:

- coprire sempre più parti
- testare quando smette di funzionare.

Diventa quasi un gioco scientifico.

Discussione con la classe
Fai riflettere su due idee chiave:

1. Ridondanza
Il QR code non contiene solo i dati essenziali: contiene anche **informazioni extra per correggere gli errori**.
2. Ricostruzione matematica
Lo scanner può **ricostruire parti mancanti** grazie a tecniche matematiche di correzione degli errori come quelle usate nei sistemi digitali.

Molto stimolante:
"Perché secondo voi i tre grandi quadrati del QR code non possono essere coperti?"
Gli studenti scoprono che servono per **orientamento e riconoscimento del codice**.

Trucco didattico divertente:
puoi anche stampare un QR code e **incollarci sopra un piccolo logo o disegno** al centro. Spesso continua a funzionare. È il motivo per cui molti QR code pubblicitari hanno **loghi dentro**.

CODICI REED-SOLOMON

L'algoritmo di **Reed-Solomon error correction** è un metodo matematico usato per **ricostruire dati che sono stati danneggiati o persi** durante una trasmissione o una lettura. È proprio uno dei sistemi che permette ai **QR code** di funzionare anche se sono graffiati o parzialmente coperti.

Il metodo di **Reed-Solomon error correction** è molto utilizzato anche in altre situazioni come ad esempio: **CD e DVD**, **trasmissioni satellitari**, **memorie digitali**, ecc. Serve ovunque i dati possano **rovinarsi durante la trasmissione** e non è possibile o conveniente la loro ritrasmissione.

Vediamo un **piccolo esempio reale di come funziona l'idea matematica dietro** a Reed-Solomon error correction usando **Python**.
Non sarà una libreria completa (quelle usate nei QR code sono molto più complesse), ma mostra **il principio matematico fondamentale: ricostruire i dati usando un polinomio anche se alcuni punti sono sbagliati o mancanti**.

L'idea alla base dell'algoritmo è questa:

1. si rappresenta il messaggio come **coefficiente di un polinomio**
2. si calcolano alcuni **punti della curva** (ridondanza), più di quelli necessari a definire il polinomio
3. se alcuni punti si perdono, **si ricostruisce il polinomio**a partire dai punti rimanenti.

ATTIVITÀ PRATICA N. 2

Esempio algoritmo Reed-Solomon semplificato in Python

Supponiamo che il messaggio sia rappresentato da questo polinomio:

$$P(x)=x^2+3x+2$$

Ora calcoliamo alcuni punti della curva.

```
import numpy as np

# polinomio: P(x) = 2 + 3x + x^2
def P(x):
    return 2 + 3*x + x**2

# punti trasmessi (ridondanza)
x_values = [0,1,2,3,4]
y_values = [P(x) for x in x_values]
```

```
print("Punti trasmessi:")
for x,y in zip(x_values,y_values):
    print(x,y)
```

Output tipico:

```
Punti trasmessi:
0 2
1 6
2 12
3 20
4 30
```

Questi sono i **dati inviati**.
Immaginiamo che durante la trasmissione **un punto si rovini**. Ora i dati ricevuti sono:

```
[2, 6, 999, 20, 30]
```

Usiamo solo alcuni punti corretti per ricostruire il polinomio originale (ne servono almeno N+1 per ricostruire perfettamente un polinomio di grado N).

```
import numpy as np

# prendiamo solo 3 punti corretti
x_sample = [0,1,3]
y_sample = [2,6,20]

coeff = np.polyfit(x_sample, y_sample, 2)

print("Coefficienti ricostruiti:", coeff)
```

Output:

```
Coefficienti ricostruiti: [1. 3. 2.]
```

che corrisponde a

$$x^2+3x+2$$

cioè il **polinomio originale**.

Come facciamo a sapere quali sono i punti errati da scartare?

Questa è **la parte davvero geniale** dell'algoritmo Reed-Solomon error correction: non solo ricostruisce i dati, ma **riesce anche a capire dove sono gli errori**.

Quando il sistema riceve i dati, fa una verifica matematica. Prende la sequenza ricevuta e la inserisce in alcune **equazioni di controllo**. I risultati di questi controlli si chiamano **sindromi**.

Se tutte le sindromi sono uguali a 0 non c'è **nessun errore**, se alcune sindromi sono ≠ 0 **ci sono errori nei dati**. È come fare un **controllo ortografico matematico**.

Se le sindromi indicano che ci sono errori, l'algoritmo costruisce una funzione speciale chiamata **polinomio localizzatore degli errori**.

Questo polinomio ha una proprietà particolare: **le sue radici indicano esattamente le posizioni degli errori**. Per trovare queste radici si usa una procedura chiamata **ricerca di Chien** (Chien search). In pratica il computer prova i possibili valori finché trova dove il polinomio diventa zero. Quando succede: quella posizione è un errore.