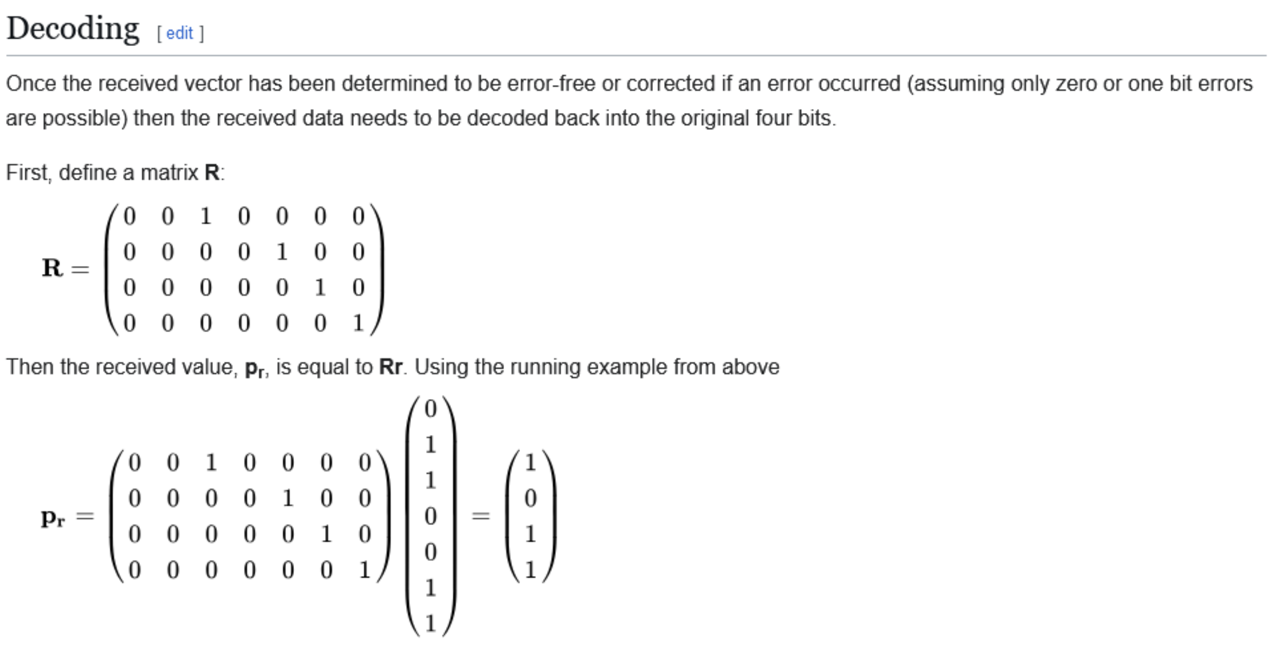
Corso di TPSIT 2024/2025 classe IV informatica - Gennaio 2025

Esercizio

Ripercorrendo l'esempio trattato a lezione, costruire una web-app (client) che, considerata la codifica di Hamming, decodifichi il codice H(7,4) in input per ottenere la sequenza di 4 bit originaria.

Implementare la seguente nota teorica:



Decoding [edit]

Once the received vector has been determined to be error-free or corrected if an error occurred (assuming only zero or one bit errors are possible) then the received data needs to be decoded back into the original four bits.

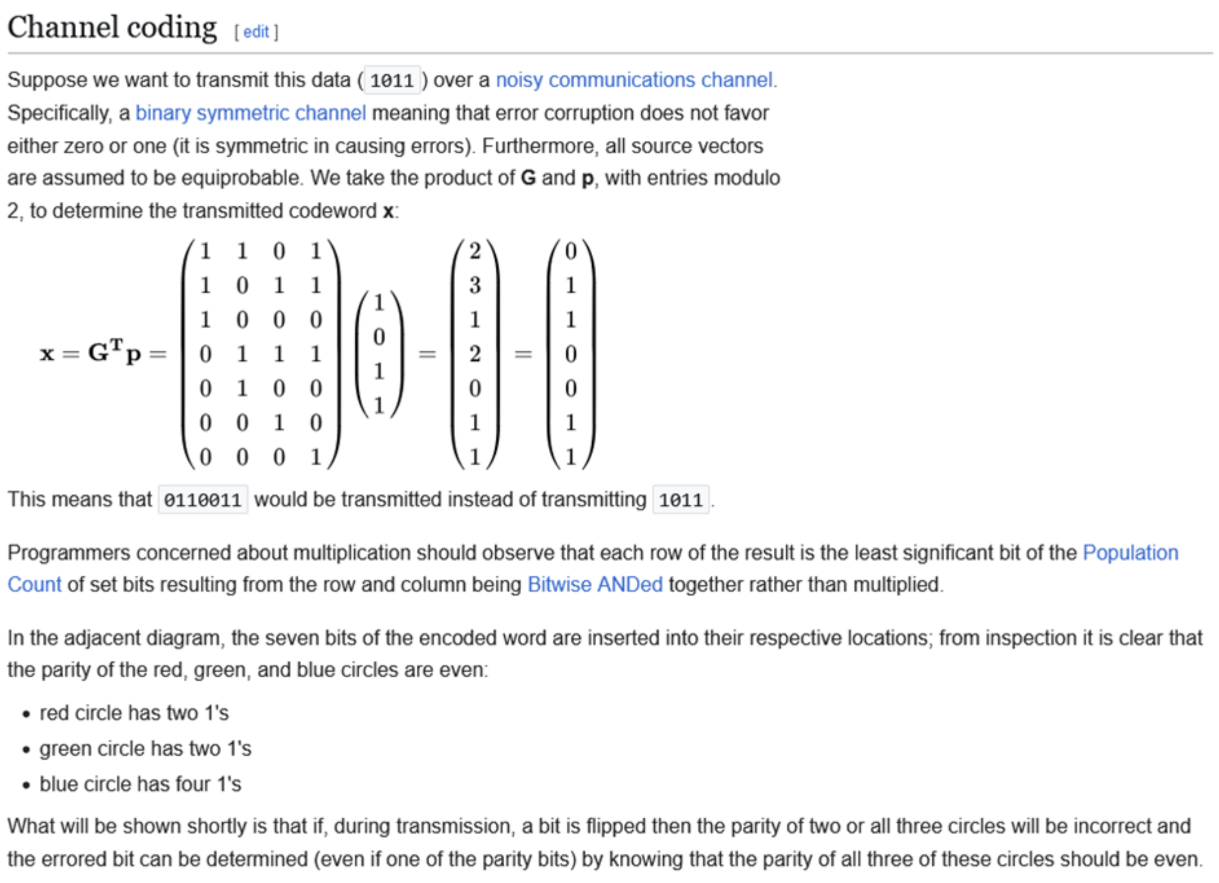
First, define a matrix **R**:

$$\mathbf{R} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Then the received value, $\mathbf{p_r}$, is equal to **Rr**. Using the running example from above

$$\mathbf{p_r} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

L'esempio di codifica, riportato di seguito, implementa la seguente nota teorica:



Channel coding [edit]

Suppose we want to transmit this data (`1011`) over a noisy communications channel. Specifically, a binary symmetric channel meaning that error corruption does not favor either zero or one (it is symmetric in causing errors). Furthermore, all source vectors are assumed to be equiprobable. We take the product of **G** and **p**, with entries modulo 2, to determine the transmitted codeword **x**:

$$\mathbf{x} = \mathbf{G^T p} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 2 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

This means that `0110011` would be transmitted instead of transmitting `1011` .

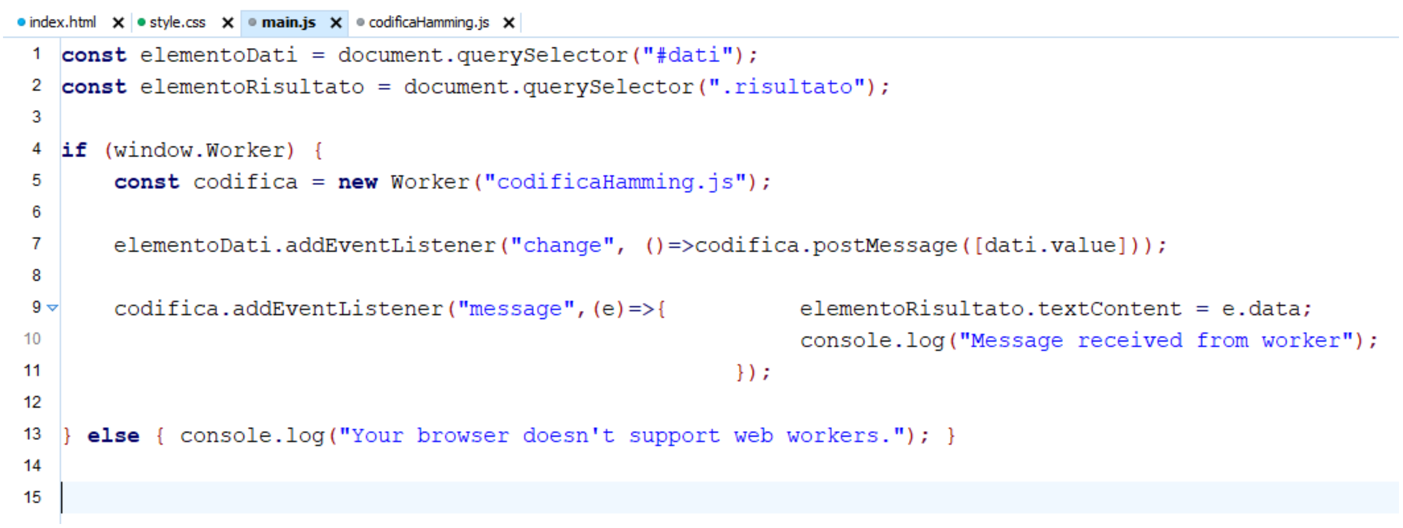
Programmers concerned about multiplication should observe that each row of the result is the least significant bit of the Population Count of set bits resulting from the row and column being Bitwise ANDed together rather than multiplied.

In the adjacent diagram, the seven bits of the encoded word are inserted into their respective locations; from inspection it is clear that the parity of the red, green, and blue circles are even:

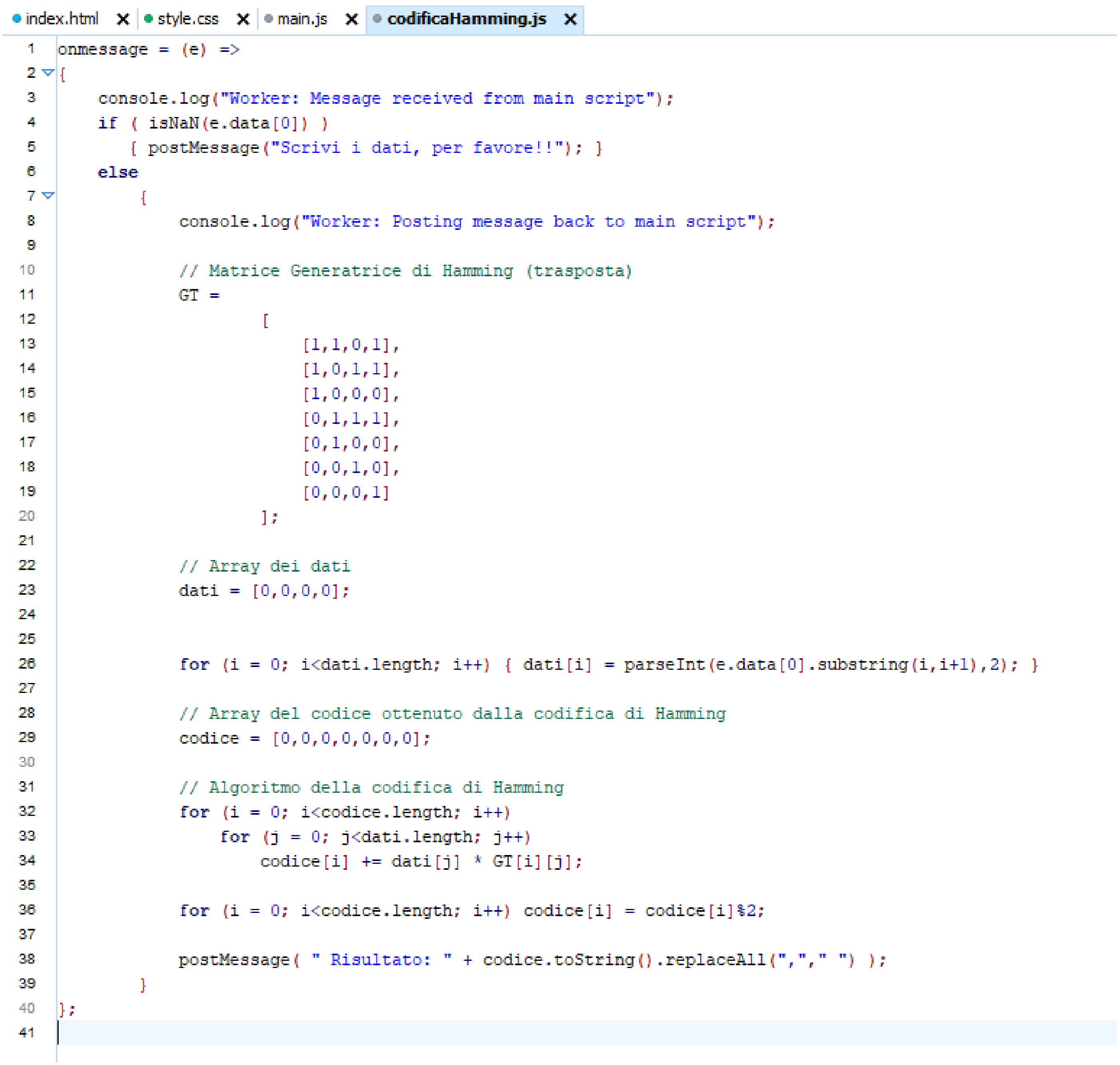
- red circle has two 1's
- green circle has two 1's
- blue circle has four 1's

What will be shown shortly is that if, during transmission, a bit is flipped then the parity of two or all three circles will be incorrect and the errored bit can be determined (even if one of the parity bits) by knowing that the parity of all three of these circles should be even.

Fonte: https://en.wikipedia.org/wiki/Hamming(7,4)#Channel_coding



```
const elementoDati = document.querySelector("#dati");
const elementoRisultato = document.querySelector(".risultato");

if (window.Worker) {
    const codifica = new Worker("codificaHamming.js");

    elementoDati.addEventListener("change",  ()=>codifica.postMessage([dati.value]));

    codifica.addEventListener("message",(e)=>{          elementoRisultato.textContent = e.data;
                                                         console.log("Message received from worker");
                                                     });

} else { console.log("Your browser doesn't support web workers."); }

```



```
onmessage = (e) =>
{
    console.log("Worker: Message received from main script");
    if ( isNaN(e.data[0]) )
       { postMessage("Scrivi i dati, per favore!!"); }
    else
        {
            console.log("Worker: Posting message back to main script");

            // Matrice Generatrice di Hamming (trasposta)
            GT =
                    [
                        [1,1,0,1],
                        [1,0,1,1],
                        [1,0,0,0],
                        [0,1,1,1],
                        [0,1,0,0],
                        [0,0,1,0],
                        [0,0,0,1]
                    ];

            // Array dei dati
            dati = [0,0,0,0];


            for (i = 0; i<dati.length; i++) { dati[i] = parseInt(e.data[0].substring(i,i+1),2); }

            // Array del codice ottenuto dalla codifica di Hamming
            codice = [0,0,0,0,0,0,0];

            // Algoritmo della codifica di Hamming
            for (i = 0; i<codice.length; i++)
                for (j = 0; j<dati.length; j++)
                    codice[i] += dati[j] * GT[i][j];

            for (i = 0; i<codice.length; i++) codice[i] = codice[i]%2;

            postMessage( " Risultato: " + codice.toString().replaceAll(","," ") );
        }
};

```