Teoria de codigos

Ulkei Szabolcs

D210871

Erasmus

Trabajo propuesto:

13. Código cíclico $C(15,7)$ generado por el polinomio $g(x) = x^8 + x^7 + x^6 + x^4 + 1$. **(Ulkei Szabolcs)**

Por primera he verificado si el polinomio esta una generador de codigo ciclo, y simultaneo he calculado el cociente de dividir entre $1 + x^{15}$ y g(x):

$$
\begin{array}{r|l}
x^8 + x^7 + x^6 + x^4 + 1 & x^{15} + 1 \\
 & \underline{x^{15} + x^{14} + x^{13} + x^{11} + x^7} \\
 & x^{14} + x^{13} + x^{11} + x^7 + 1 \\
 & \underline{x^{14} + x^{13} + x^{12} + x^{10} + x^6} \\
 & x^{12} + x^{11} + x^{10} + x^7 + x^6 + 1 \\
 & \underline{x^{12} + x^{11} + x^{10} + x^8 + x^4} \\
 & x^8 + x^7 + x^6 + x^4 + 1 \\
 & \underline{x^8 + x^7 + x^6 + x^4 + 1} \\
 & -
\end{array}
\qquad x^7 + x^6 + x^4 + 1
$$

$\mapsto$ g(x) está un generador de codigo

$h(x) = x^7 + x^6 + x^4 + 1$

Ahora con uno generador valido y tuve de construir las matrizes generador y comprobacion de paridad:

```
Matriz Generador Codigo:
1 0 0 0 1 0 1 1 0 0 0 0 0 0 0
0 1 0 0 0 1 0 1 1 0 0 0 0 0 0
0 0 1 0 0 0 1 0 1 1 0 0 0 0 0
0 0 0 1 0 0 0 1 0 1 1 0 0 0 0
0 0 0 0 1 0 0 0 1 0 1 1 0 0 0
0 0 0 0 0 1 0 0 0 1 0 1 1 0 0
0 0 0 0 0 0 1 0 0 0 1 0 1 1 0

Matriz Comprobacion Paridad:
1 0 0 1 0 1 1 0 0 0 0 0 0 0 0
0 1 0 0 1 0 1 1 0 0 0 0 0 0 0
0 0 1 0 0 1 0 1 1 0 0 0 0 0 0
0 0 0 1 0 0 1 0 1 1 0 0 0 0 0
0 0 0 0 1 0 0 1 0 1 1 0 0 0 0
0 0 0 0 0 1 0 0 1 0 1 1 0 0 0
0 0 0 0 0 0 1 0 0 1 0 1 1 0 0
0 0 0 0 0 0 0 1 0 0 1 0 1 1 0
```

$$G = \begin{pmatrix} 1 & g_1 & g_2 & \cdots & g_{n-k-1} & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & g_1 & \cdots & g_{n-k-2} & g_{n-k-1} & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & g_{n-k-3} & g_{n-k-2} & g_{n-k-1} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & g_1 & g_2 & g_3 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & g_1 & g_2 & \cdots & g_{n-k-1} & 1 \end{pmatrix}.$$

$$: H = \begin{pmatrix} h_k & h_{k-1} & \cdots & h_1 & h_0 & 0 & \cdots & 0 \\ 0 & h_k & \cdots & h_2 & h_1 & h_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \end{pmatrix}$$

En fine para generar las todas palabras del codigo ciclo, primera genero las todas polinomios con un grado < n. Esto he hecho con simplemente generara polinomios similares con numerar en binar: 0 -> 01 -> 10 -> 11 ….

```
polinomios = (int**)malloc(nr_polinomios * sizeof(int));

polinomios[0] = (int*)calloc(k, sizeof(int));
for(int i = 1; i < nr_polinomios; i++){
    polinomios[i] = (int*)malloc(k * sizeof(int));
    int carry = 1;
    for(int j = 0; j < k; j++){
        if(
            (polinomios[i - 1][j] == 1 && carry == 0) ||
            (polinomios[i - 1][j] == 0 && carry == 1)
        ){
            polinomios[i][j] = 1;
            carry = 0;
        }
```

```
        else if(
            polinomios[i - 1][j] == 0 && carry == 0
        ){
            polinomios[i][j] = 0;
            carry = 0;
        }
        else{
            polinomios[i][j] = 0;
            carry = 1;
        }
    }
}
```

Despues de eso, multiplicando con el generador del codigo, obtenemos las palabras del codigo:

```
Palabras:{
000000000000000, 100010111000000, 010001011100000, 110011100100000,
001000101110000, 101010010110000, 011001110010000, 111011001010000,
000100010111000, 100110101111000, 010101001011000, 110111110011000,
001100111001000, 101110000001000, 011101100101000, 111111011101000,
000010001011100, 100000110011100, 010011010111100, 110001101111100,
001010100101100, 101000011101100, 011011111001100, 111001000001100,
000110011100100, 100100100100100, 010111000000100, 110101111000100,
001110110010100, 101100001010100, 011111101110100, 111101010110100,
000001000101110, 100011111101110, 010000011001110, 110010100001110,
001001101011110, 101011010011110, 011000110111110, 111010001111110,
000101010010110, 100111101010110, 010100001110110, 110110110110110,
001101111100110, 101111000100110, 011100100000110, 111110011000110,
000011001110010, 100001110110010, 010010010010010, 110000101010010,
001011100000010, 101001011000010, 011010111100010, 111000000100010,
000111011001010, 100101100001010, 010110000101010, 110100111101010,
001111110111010, 101101001111010, 011110101011010, 111100010011010,
000000100010111, 100010011010111, 010001111110111, 110011000110111,
001000001100111, 101010110100111, 011001010000111, 111011101000111,
000100110101111, 100110001101111, 010101101001111, 110111010001111,
001100011011111, 101110100011111, 011101000111111, 111111111111111,
000010101001011, 100000010001011, 010011110101011, 110001001101011,
001010000111011, 101000111111011, 011011011011011, 111001100011011,
000110111110011, 100100000110011, 010111100010011, 110101011010011,
001110010000011, 101100101000011, 011111001100011, 111101110100011,
000001100111001, 100011011111001, 010000111011001, 110010000011001,
001001001001001, 101011110001001, 011000010101001, 111010101101001,
000101110000001, 100111001000001, 010100101100001, 110110010100001,
001101011110001, 101111100110001, 011100000010001, 111110111010001,
000011101100101, 100001010100101, 010010110000101, 110000001000101,
001011000010101, 101001111010101, 011010011110101, 111000100110101,
000111111011101, 100101000011101, 010110100111101, 110100011111101,
001111010101101, 101101101101101, 011110001001101, 111100110001101
}
```