Novedades de software/New softwares

NUEVO ALGORITMO PARA LA MULTIPLICACION DE MATRICES BOOLEANAS

P. Freyre[1] y N.Díaz
Facultad de Matemática y Computación, Universidad de La Habana, Cuba.

ABSTRACT
In this paper is shown an algorithm about the multiplication of nxn invertibles boolean square matrixes. The algorithm has as novelty that the operations that this algorithm perform are addition and multiplications in the finite field GF(2) and multiplications between polynomials modulo a primitive polynomial, the polynomials belongs to GF(2) [x].

MSC: 65F05

KEYWORDS: matrices fast multiplication, matrices on a finite field, algorithm for matrices.

RESUMEN
En el presente artículo se muestra un algoritmo para la multiplicación de matrices booleanas cuadradas nxn invertibles. El algoritmo tiene como novedad que las operaciones que se realizan son sumas y multiplicaciones en el campo finito *GF(2)* y multiplicación de polinomios módulo un polinomio primitivo, los polinomios pertenecen a *GF(2)[x]*.

1. INTRODUCCIÓN

La búsqueda de nuevos y eficientes algoritmos para multiplicar matrices es un área de constante investigación por parte de la comunicada científica. En 1967 S. Winograd descubre que la multiplicación de matrices puede realizarse con menos multiplicaciones a las del método tradicional. En 1968 V. Strassen encuentra una manera más eficiente para multiplicar dos matrices 2x2 (Strassen V. (1969)y Knuth E. D. (1981)). Los trabajos para mejorar la complejidad en la multiplicación de matrices han continuado y nuevas propuestas de algoritmos se han realizado. Strassen V. (1969), Knuth E. D. (1981), Schonhage A. (1981), Pan V. Y. (1984), Coppersmith D. and Winograd S. (1990), Highman N. J. (1996) y Majan M. and Vinay V. (1997).
El presente trabajo tiene como objetivo exponer, programado en Mathematica, un nuevo algoritmo para multiplicar matrices booleanas cuadradas nxn e invertibles que tiene como novedad, que las operaciones que se realizan son sumas y multiplicaciones en GF(2) y multiplicaciones de polinomios módulo un polinomio primitivo, los polinomios pertenecen a GF(2)[x] Freyre P, Díaz N y Morgado E. R. (2009).
En el trabajo se dan 4 ejemplos de corrida del algoritmo. Los polinomios primitivos que se utilizan se encuentran en Peterson W.W. y Weldon J. E. (1972), Golomb W. S. (1982) y Lidl R. y Niederreiter H. (1994).

2. MULTIPLICACION DE MATRICES.

En el programa tenemos que:

n – Es el tamaño de la matriz.

lpp – Es la lista de polinomios primitivos $g_i(x) \in GF(2)[x]$, $i \in \{1 \ldots n\}$, a utilizarse en el algoritmo, representados en forma descendente según su grado, y se calculan con anterioridad. Los mismos pueden ser seleccionados arbitrariamente.

[1] pfreyre@matcom.uh.cu

h – Matriz de tamaño ninvertibles.

t – Matriz de tamaño n invertibles.

m – Matriz resultante de multiplicar *h por t, m= h × t.*

ALGORITMO PARA LA MULTIPLICACION DE MATRICES

Programación del algoritmo.

```
Clear[Creavbc]
Creavbc[n_,i_,v_,vbc_,lpp_]:=
Block[{x=0,t,z,y=PadLeft[{},n]},
   z =lpp[[i]][Take[vbc[[i]],{i,n}]];
   t = lpp[[i]][Take[y,{i,n}]];
   If[TrueQ[z[[1]] □ t[[1]]],t=lpp[[i]][Take[v,{i,n}]]*(z^-1);
If[TrueQ[t□0],t=lpp[[i]][Take[y,{i,n}]]];
    x=lpp[[1]][Take[v,{1,i-1}]] -
lpp[[1]][Take[vbc[[i]],{1,i-1}]]*lpp[[1]][t[[1]][[{1}]]];
    ];
If[TrueQ[x□0],x=lpp[[1]][y]];
Return[Join[Take[x[[1]],{1,i-1}],t[[1]]]];
   ]

Clear[Fpolinomial]
Fpolinomial[n_,m_,lpp_]:=
Block[{vbc={},i,j,vec,y=PadLeft[{},n]},
For[j=1, j<=n,j++,
i=0;vec=m[[j]];
While[(i=i+1)<j,vec=Creavbc[n,i,vec,vbc,lpp];
If[Take[vec,{i,n}] □Take[y,{i,n}],Print[" No Inversible "];
        Return[y]]
      ];
If[Take[vec,{i,n}] □Take[y,{i,n}],Print[" No Inversible "];
      Return[y]];
AppendTo[vbc,vec]
    ];
Return[vbc];
   ]

Clear[Mulmatriz]
Mulmatriz[n_,h_,t_,lpp_]:=
Block[{m={},vbc,j,vec},
vbc=Fpolinomial[n,t,lpp];
If[TrueQ[vbc□PadLeft[{},n]],
Print["Por ser no Inversible no se puede Multiplicar por este metodo."];
Return[vbc]];
For[j=1, j<=n,j++,
vec=h[[j]];Do[vec=Lbi[n,n - i,vec,vbc,lpp],{i,0,n-1}];
AppendTo[m,vec]
    ];
   Return[m];
]
```

Ejemplo 1: Dados los polinomios primitivos:*x^3 + x^2 + 1, x^2 + x + 1, x + 1*

```
<<Algebra`FiniteFields`
lpp = {GF[2, {1, 0, 1, 1}], GF[2, {1, 1, 1}],GF[2, {1, 1}]}
h = {{0, 1, 1}, {1, 0, 0}, {0, 0, 1}}
t = {{1, 0, 0}, {0, 0, 1}, {1, 1, 0}}
m = Mulmatriz[3, h, t, lpp]
MatrixForm[%]
```

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

Ejemplo 2: Dados los polinomios primitivos:*x^4 + x + 1, x^3 + x^2 + 1, x^2 + x + 1, x + 1.*

```
<<Algebra`FiniteFields`
lpp = {GF[2,{1,1,0,0,1}],GF[2,{1,0,1,1}],GF[2,{1,1,1}],GF[2,{1,1}]}
h = {{0,1,0,1},{0,1,0,0},{0,1,1,1},{1,1,1,1}}
t = {{1,0,0,0},{1,0,1,1},{0,1,0,0},{0,1,1,0}}
m = Mulmatriz[4, h, t, lpp]
MatrixForm[%]
```

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

Ejemplo 3: Dados los polinomios primitivos:.*x^5 + x^2 + 1, x^4 + x + 1, x^3 + x^2 + 1, x^2 + x + 1, x + 1*

```
<<Algebra`FiniteFields`
lpp = {GF[2, {1, 0, 1, 0, 0, 1}], GF[2, {1, 1, 0, 0, 1}], GF[2, {1, 0, 1, 1}],
GF[2, {1, 1, 1}], GF[2, {1, 1}]}
h = {{0, 0, 1, 1, 0}, {1, 1, 0, 1, 1}, {1, 0, 1, 1, 1}, {0, 0, 0, 0, 1}, {0, 1, 0, 0, 1}}
t = {{0, 0, 0, 1, 0}, {0, 1, 0, 1, 1}, {0, 1, 1, 0, 0}, {0, 1, 1, 0, 1}, {1, 0, 1, 1, 0}}
m = Mulmatriz[5, h, t, lpp]
MatrixForm[%]
```

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

Ejemplo 4.: Dados los polinomios primitivos**:** *x^6 + x + 1, x^5 + x^2 + 1, x^4 + x + 1, x^3 + x^2 + 1, x^2 + x + 1, x + 1*

```
<<Algebra`FiniteFields`
lpp = {GF[2, {1, 1, 0, 0, 0, 0, 1}], GF[2, {1, 0, 1, 0, 0, 1}], GF[2, {1, 1, 0, 0, 1}],
GF[2, {1, 0, 1, 1}], GF[2, {1, 1, 1}], GF[2, {1, 1}]}
h = {{1, 0, 1, 0, 0, 0}, {0, 1, 0, 0, 0, 0}, {1, 0, 0, 0, 0, 1}, {0, 1, 0, 0, 1, 1},
{0, 1, 1, 1, 0, 1}, {0, 0, 1, 1, 0, 0}}
t = {{0, 1, 0, 1, 1, 0}, {1, 1, 1, 0, 1, 1}, {1, 0, 0, 0, 1, 1}, {0, 0, 0, 1, 0, 1},
{0, 0, 1, 1, 1, 1}, {1, 0, 0, 0, 0, 0}}
m = Mulmatriz[6, h, t, lpp]
MatrixForm[%]
```

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

RECEIVED NOVEMBER, 2012
REVISED FEBRUARY, 2013

REFERENCIAS

[1] COPPERSMITH, D. and WINOGRAD, S. (1990):**Matrix multiplication via Arithmetic progressions.** J. Symbolic Comput. 9, 251_280.

[2] FREYRE, P, DÍAZ, N. y MORGADO, E. R. (2009): **Some algorithms related to matrices with entries in a finite field.** Journal of Discrete Mathematical Sciences & Cryptography. Vol. 12, No. 5, pp. 509–519. India.

[3] GOLOMB, W. S. (1982): **Shift Register Sequences**. Aegean Park Press. California.

[4] HIGHMAN, N. J. (1996): **Accuracy and Stability of Numerical Algorithms.** SIAM. Philadelphia.

[5] KNUTH, E. D. (1981): **The Art of Computer Programming**. Vol 2. Addison – Wesley. 2da ed. , N. York.

[6] LIDL, R. Y. And NIEDERREITER, H. (1994): **Introduction to Finite Fields and their Applications**. Cambridge University. New York.

[7] MAJAN, M and VINAY V. (1997): **Determinant: Combinatorics, Algorithms, and Complexity**. http://cjtcs.cs.uchicago.edu/articles.

[8] PAN, V. Y.(1984): **How can we speed-up matrix multiplication?**, SIAM. Rev. 26, 393_415.

[9] PETERSON, W.W and WELDON, J. E. (1972): **Error – Correcting Codes,** 2ed. John Wiley and Sons, Inc. New York.

[10] SCHONHAGE, A. (1981): **Partial and total matrix multiplication.** SIAM J. Comput.10, 434_456.

[11] STRASSEN, V. (1969): **Gaussian elimination is not optimal.** Num Math. 13, 354_356.