Implementación y Análisis de Código

Kevin Jhomar Sanchez Sanchez

I. Algoritmo de MCD

I-A. Código

```
1. ZZ euclides(ZZ a, ZZ b){
2.     ZZ r, d;
3.     r = rsta = 1;
4.     while(r > 0){
5.         d = r;
6.         r = mod(a, b);
7.         a = b;
8.         b = r;
9.     }
10.    return d;
11. }
```

I-B. Análisis

Bits	**Tiempo** (ms)	**Vueltas**
10	0.111	10
50	0.181	35
100	0.229	63
200	0.480	124
400	0.668	232
600	1.147	297
800	1.321	461
1023	1.542	465

I-C. Seguimiento del algoritmo

	a	**b**	**r**	**d**
Inicio	1827	1024	1	1
1	1024	803	803	803
2	803	221	221	221
3	221	140	140	140
4	140	81	81	91
⋮	⋮	⋮	⋮	⋮
9	15	7	7	7
10	7	1	1	1

Donde:

a : *Entrada*
b : *Entrada*
r : *Residuo*
d : *Es el MCD*

II. Algoritmo de Resto Mínimo

II-A. Código

```
1. ZZ resto_minimo(ZZ a, ZZ b){
2.     ZZ c, d, r;
3.     if(a == 0){
4.         return b;
5.     }else{
6.         while(b != 0){
7.             r = mod(a, b);
8.             a = b;
9.             b = r;
10.        }
11.    }
12.    return a;
13. }
```

II-B. Análisis

Bits	**Tiempo** (ms)	**Vueltas**
10	0.071	10
50	0.115	35
100	0.114	63
200	0.124	124
400	0.348	232
600	0.300	297
800	0.660	461
1023	0.671	465

II-C. Seguimiento del algoritmo

	a	**b**	**r**
Inicio	1827	1024	803
1	1024	803	221
2	803	221	140
3	221	140	59
4	81	59	22
⋮	⋮	⋮	⋮
8	15	7	1
9	7	1	0

Donde:

a : *Entrada*
b : *Entrada*
r : *Residuo*

III. Algoritmo LSBGCD

III-A. Código

```
1. ZZ LSBGCD(ZZ a, ZZb){
2.     ZZ t = (ZZ)0; int s = 0;
3.     while(b != (ZZ)0){
4.         s = 0;
5.         while((b << s) <= a) s++;
6.         --s;
7.
8.         t = min(a-(b << s),(b << s+1)-a);
9.         a = b; b = t;
10.        if(b > a){
11.            ZZ temp = a;
12.            a = b;
13.            b = temp;
14.        }
15.    }
16.    return a;
17. }
```

III-B. Análisis

Bits	**Tiempo** (ms)	**Vueltas**
10	0.092	9
50	0.227	39
100	0.348	69
200	0.676	136
400	1.331	270
600	3.149	357
800	2.769	516
1023	53.765	587

III-C. Seguimiento del algoritmo

	a	**b**	**s**	**t**
Inicio	1827	1024	0	0
1	1024	221	0	221
2	221	140	2	140
3	140	59	0	59
4	59	22	1	22
⋮	⋮	⋮	⋮	
7	7	1	1	1
8	1	1	2	1

Donde:
a : $Entrada$
b : $Entrada$
s : $Incrementador$
t : $VariableTemporal$

IV. Algoritmo de MCD Binario

IV-A. Código

```
1. ZZ euclides_binary(ZZ a, ZZ b){
2.     int i;
3.     if(a == 0) return b;
4.     else if(b == 0) return a;
5.     else if(b == a) return a;
6.     else{
7.         for(i = 0;(a&1) == 0 && (b&1) ==
0;i++){
8.             a = a >> 1;
9.             b = b >> 1;
10.        }
11.
12.        while((a & 1) == 0) a = a >> 1;
13.        while(b != 0){
14.        while((b & 1) == 0) b = b >> 1;
15.            if(a > b){
16.                ZZ t = b;
17.                b = a;
18.                a = t;
19.            }
20.            b = b - a;
21.        }
22.        return a << i;
23.    }
24. }
```

IV-B. Análisis

Bits	**Tiempo** (ms)	**Vueltas**
10	0.083	22
50	0.107	110
100	0.334	223
200	0.426	417
400	0.363	850
600	0.386	1242
800	0.698	1641
1023	0.977	2136

NOTA: Se han aplicado las propiedades binarias.

- Cuando 'a' y 'b' son pares.
  $2 * MCD(a/2, b/2)$
- Cuando 'a' es par y 'b' es impar o viceversa.
  $MCD(a/2, b)$
- Cuando 'a' y 'b' son impares o viceversa.
  $MCD(a - b, b)$

IV-C. Seguimiento del algoritmo

	a	b	t
Inicio	1827	1024	0
1	1024	803	1024
2	803	221	803
4	221	582	221
5	221	291	221
6	221	70	221
7	221	35	221
⋮	⋮	⋮	⋮
21	1	1	1
22	1	0	1

Donde:

a : *Entrada*

b : *Entrada*

t : *Variable Temporal*

V. Algoritmo de Euclides Extendido

V-A. Código

```
1. vector<ZZ> MCD_extended(ZZ a, ZZ b){
2.     ZZ r, d, q, x, y;
3.     x = y = r = d = 0;
4.     if(a != 0 & b == 0){
5.         x = 1; y = 0; d = a;
6.     }else if(a == 0&b != 0){
7.         x = 0; y = 1; d = b;
8.     }else{
9.         ZZ x1 = (ZZ)0, x2 = (ZZ)1,
10.            y1 = (ZZ)1, y2 = (ZZ)0;
10.        while(b > 0){
11.            q = a/b;
12.            r = a - q*b;
13.            x = x2 - q*x1;
14.            y = y2 - q*y1;
15.            a = b;
16.            b = r;
17.            x2 = x1;
18.            x1 = x;
19.            y2 = y1;
20.            y1 = y;
21.        }
22.        d = a;
23.        x = x2;
24.        y = y2;
25.    }
26.    vector<ZZ> n;
26.    n.push_back(d); n.push_back(x); n.push_back(y);
27.    return n;
28.}
```

V-B. Análisis

Bits	**Tiempo** (ms)	**Vueltas**
10	0.152	10
50	0.278	35
100	0.406	63
200	1.003	124
400	1.260	232
600	0.810	297
800	1.957	461
1023	1.576	465

V-C. Seguimiento del algoritmo

	q	x	y	x1	x2	y1	y2
Inicio	0	0	0	0	1	1	0
1	1024	1	-1	1	0	-1	1
2	803	-1	2	-1	1	2	-1
3	221	4	-7	4	-1	-7	2
4	140	-5	9	-5	4	9	-7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
9	15	-51	91	-51	37	91	-66
10	7	139	-248	139	-51	-248	91

Donde:

q : *Cociente*

x : *Residuo*

y : *Residuo*

$x1$: *Dividendo* $x2$: *Divisor*

$y1$: *Dividendo* $y2$: *Divisor*

VI. Algoritmo de MCD Binario Extendido

VI-A. Código

```
1. vector < ZZ > binary_extended(ZZ x, ZZ y){
2.     ZZ res;
3.     int g = 0;
4.     if(x == 0) res = y;
5.     else if(y == 0) res = x;
6.     else if(y == x) res = x;
7.     else{
8.         for(int i = 0;(x&1) == 0 && (y&1) == 0;i++){
9.             x = x >> 1;
10.            y = y >> 1;
11.            g++;
12.        }
13.
14.        ZZ u = x, v = y, A = (ZZ)1, B = (ZZ)0, C = (ZZ)0, D = (ZZ)1;
15.        while(u != 0){
16.            while((u &1) == 0){
17.                u = u >> 1;
18.                if(mod(A,(ZZ)2) == 0 and mod(B,(ZZ)2) == 0){
19.                    A = A >> 1;
20.                    B = B >> 1;
21.                }else{
22.                    A = (A + y) >> 1;
23.                    B = (B - x) >> 1;
24.                }
25.            }
26.            while((v & 1) == 0){
27.                v = v >> 1;
28.                if(mod(C,(ZZ)2) == 0 and mod(D,(ZZ)2) == 0){
29.                    C = C >> 1;
30.                    D = D >> 1;
31.                }else{
32.                    C = (C + y) >> 1;
33.                    D = (D - x) >> 1;
34.                }
35.            }
36.
37.            if(u >= v){
38.                u = u - v;
39.                A = A - C;
40.                B = B - D;
41.            }else{
42.                v = v - u;
43.                C = C - A;
44.                D = D - B;
45.            }
46.        }
47.        if(u == 0){
```

```
48.             x = C;
49.             y = D;
50.         }
51.
52.         res = v << g;
53.     }
54.     vector < ZZ > n;
55.     n.push_back(res); n.push_back(x); n.push_back(y);
56.     return n;
57. }
```

VI-B. Análisis

Bits	**Tiempo** (ms)	**Vueltas**
10	0.306	6
50	0.707	33
100	0.443	78
200	2.070	138
400	4.715	282
600	4.706	431
800	8.185	541
1023	10.837	724

NOTA: Se han aplicado las propiedades binarias.

- Cuando 'a' y 'b' son pares.
  $2 * MCD(a/2, b/2)$
- Cuando 'a' es par y 'b' es impar o viceversa.
  $MCD(a/2, b)$
- Cuando 'a' y 'b' son impares o viceversa.
  $MCD(a - b, b)$

VI-C. Seguimiento del algoritmo

	x	**y**	**u**	**v**	**A**	**B**	**C**	**D**
Inicio	1827	1024	1827	1024	1	0	0	1
1	1827	1024	1827	1	-138	248	139	-248
2	1827	1024	912	1	-208	372	139	-248
3	1827	1024	56	1	616	-1099	139	-248
4	1827	1024	6	1	834	-1488	139	-248
5	1827	1024	2	1	278	-496	139	-248
6	139	-248	0	1	0	0	139	-248

Donde:
$x,\ y\ :\ Son\ los\ valores\ inicial.$
$u,\ v\ :\ Valores\ iniciales\ que\ se\ modifica\ durante\ el\ programa$
$A, B, C, D\ :\ Son\ los\ valores\ fijados\ para\ hallar\ el\ extendido.$

VII. ALGORITMO MCD DE LEHMER

VII-A. Código

```
1. ZZ lehmer_mcd(ZZ A, ZZ B, ZZ p, ZZ system){
2.     ZZ a0, a1, b, W, q0, q1, r, u0, u1, v0, v1, R, T;
3.     W = exp(system, p);
4.     int h;
5.     while(B >= W){
6.         conv(h, NumBits(B) - p + 1);
7.         a0 = A >> h;
8.         a1 = B >> h;
9.         u0 = (ZZ)1; u1 = (ZZ)0; v0 = (ZZ)0; v1 = (ZZ)1;
10.        while((a1 + u1 != (ZZ)0)and(a1 + v1 != (ZZ)0)){
11.            q0 = (a0 + u0)/(a1 + u1);
12.            q1 = (a0 + v0)/(a1 + v1);
13.
14.            if(q0 != q1)
15.                break;
16.
17.            r = a0 - q0 * a1; a0 = a1; a1 = r;
18.            r = u0 - q0 * u1; u0 = u1; u1 = r;
19.            r = v0 - q0 * v1; v0 = v1; v1 = r;
20.        }
21.
22.        if (v0 == (ZZ)0){
23.            R = mod(A, B);
24.            A = B;
25.            B = R;
26.        }else{
27.            R = u0 * A + v0 * B;
28.            T = u1 * A + v1 * B;
29.            A = R;
30.            B = T;
31.        }
32.        if(B == 0) return A;
33.    }
34.    A = euclides(A, B);
35.    return A;
36. }
```

VII-B. Análisis

Bits	**Tiempo** (ms)	**Vueltas**
10	0.092	1
50	0.292	30
100	0.912	58
200	1.201	122
400	0.882	238
600	1.134	315
800	2.321	490
1023	4.095	487

VII-C. Seguimiento del algoritmo

	A	**B**	**a0**	**a1**	**q0**	**q1**	**r**	**u0**	**u1**	**v0**	**v1**	**R**	**T**
Inicio	986523541	894612341	0	0	0	0	0	0	0	0	0	0	0
1	894612341	91911200	3	3	1	0	0	1	0	0	1	91911200	0
2	91911200	67411541	26	2	13	8	0	1	0	0	1	67411541	0
3	67411541	24499659	2	2	1	0	0	1	0	0	1	24499659	0
4	24499659	18412223	8	2	4	2	0	1	0	0	1	18412223	0
5	18412223	6087436	2	2	1	0	0	1	0	0	1	6087436	0
6	6087436	149915	8	2	4	2	0	1	0	0	1	149915	0
7	149915	90836	92	2	46	30	0	1	0	0	1	90836	0
8	90836	59079	4	2	2	1	0	1	0	0	1	59079	0
9	59079	31757	5	3	2	1	0	1	0	0	1	31757	0
10	31757	27322	7	3	2	1	0	1	0	0	1	27322	0
11	27322	4435	3	3	1	0	0	1	0	0	1	4435	0
A partir de Aqui se realiza el Algoritmo de Euclides													

Donde:
$A,\ B\ :\ Entradas$
$a1,\ a2:\ Son\ las\ entradas\ que\ han\ sido\ desplazadas\ a\ la\ derecha$
$r\ :\ Residuo$
$u0,\ u1,\ v0,\ v1:\ Variables\ Temporales\ del\ residuo$
$R,\ T\ :\ Variables\ Temporales$

VIII. Conclusión

Bueno como hemos visto cada algoritmo tiene lo suyo ya que algunos mientras más bits tengan las variables de entrada dan menos vueltas y se demoran menos, otros es todo lo contrario mientras más bits tengan las variables de entrada se demoran más. Pienso que cada algoritmo se tiene que usar para un caso en específico del que se esta desarrollando. A continuación presento las tablas de análisis finales de todos los algoritmos visto.

Tabla de Análisis por Tiempo de Respuesta (ms)

Bits	**MCD**	**Resto Mínimo**	**LSBGCD**	**Binario**	**MCD Extendido**	**Binario Extendido**	**Lehmer**
10	0.111	0.071	0.092	0.083	0.152	0.306	0.092
50	0.181	0.115	0.227	0.107	0.278	0.707	0.292
100	0.229	0.114	0.348	0.334	0.406	0.443	0.912
200	0.480	0.124	0.676	0.426	1.003	2.070	1.201
400	0.668	0.348	1.331	0.363	1.260	4.715	0.882
600	1.147	0.300	3.159	0.386	1.520	4.706	1.134
800	1.321	0.660	2.769	0.698	1.957	8.185	2.321
1023	1.542	0.671	53.765	0.977	1.576	10.837	4.095

Tabla de Análisis por Número de Iteraciones

Bits	**MCD**	**Resto Mínimo**	**LSBGCD**	**Binario**	**MCD Extendido**	**Binario Extendido**	**Lehmer**
10	10	10	9	22	10	6	1
50	35	35	39	110	35	33	30
100	63	63	69	223	63	78	58
200	124	124	136	417	124	138	122
400	232	232	270	850	232	282	238
600	297	297	357	1242	297	431	215
800	461	461	516	1641	461	541	490
1023	465	465	587	2136	465	724	487