Análisis de MCD y MCD Binario

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**
2^{10}	0.111	10
2^{50}	0.181	35
2^{100}	0.229	63
2^{200}	0.480	124
2^{400}	0.668	232
2^{600}	1.147	297
2^{800}	1.321	461
2^{1023}	1.542	465

NOTA: No se han aplicado ningunas de las propiedades.

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 MCD Binario

II-A. Código

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

II-B. Análisis

Bits	**Tiempo** (ms)	**Vueltas**
2^{10}	0.197	25
2^{50}	0.719	105
2^{100}	1.265	222
2^{200}	7.480	416
2^{400}	21.166	848
2^{600}	42.771	1241
2^{800}	53.361	1640
2^{1023}	67.213	2135

NOTA: Se han aplicado las propiedades binarias.

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

II-C. Seguimiento del algoritmo

	a	b	d	Propiedad
Inicio	1827	1024	0	...
1	1827	512	0	P2
2	1827	256	0	P2
3	1827	128	0	P2
4	1827	64	0	P2
5	1827	32	0	P2
6	1827	16	0	P2
7	1827	8	0	P2
8	1827	4	0	P2
9	1827	2	0	P2
10	1827	1	0	P2
11	913	1	0	P2
12	912	1	0	P3
⋮	⋮	⋮	⋮	⋮
25	2	1	0	P2
26	1	1	1	...

Donde:
a : $Entrada$
b : $Entrada$
d : $Es\ el\ MCD$
$P2$: $Propiedad\ Dos$
$P3$: $Propiedad\ Tres$

III. Algoritmo de Euclides Extendido

III-A. Código

```
1. ZZ euclides_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.    cout << x << " " << y << endl;
27.    return d;
28.}
```

III-B. Análisis

Bits	Tiempo (ms)	Vueltas
2^{10}	0.152	10
2^{50}	0.278	35
2^{100}	0.406	63
2^{200}	1.003	124
2^{400}	1.260	232
2^{600}	0.810	297
2^{800}	1.957	461
2^{1023}	1.576	465

III-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*

IV. Conclusión

El análisis realizado me a dejado muy desconcertado porque a primera vista pareciera que el euclides binario sería muy rápido pero los resultamos me mustran pero no, es mucho más lento y con un costo mucho más alto de lo que se suponía que debería ser, quizá mi código esta mal. Esto significa que el mejor código para realizar el MCD es el básico.