TRABAJO DE INVESTIGACIÓN E IMPLEMENTACIÓN: ALGORITMO DE EUCLIDES Y ALGORITMO EXTENDIDO DE EUCLIDES

Integrantes:
Becerra Sipiran, Cledy Elizabeth
Oviedo Sivincha, Massiel
Villanueva Borda, Harold Alejandro

Docente:
Dc. Ana Maria Cuadros Valdivia

Curso:
Álgebra Abstracta

Departamento de Ciencia de la Computación
Universidad Católica San Pablo

Semestre 2021 - III
Arequipa - Perú

Introducción

- Investigar las diversas variantes del Algoritmo de Euclides y el Algoritmo de Euclides Extendido.
- Analizar algoritmos y evaluar su eficiencia.
- Encontrar los algoritmos más eficientes entre los investigados.

Algoritmo para hallar MCD

Euclides Clásico

Euclides Menor Resto

Bishop MCD

MCD Binario

MCD Lehmer

El Mejor MCD

Teorema de la división: A = B*q +r

Identidades de Bezout:

Si A, B son dos enteros no ambos cero, existen x,y ∈ Z (posiblemente no únicos) tales que A(x) + B(y) = gcd(A, B)

Sean 0 < B < A enteros. El algoritmo de Euclides consiste en computarizar (A, B): (A0, A1,..., An , An+i) definido por las relaciones:

A0 = A, A1 = B, Ai+2 = A; mod Ai+1, An+1 = 0. Es sabido que An = gcd(A, B).

En el extendido, las primeras y segundas secuencias simultáneas (U0, U1, ... , Un+1) y (V0, V1, ... , Vn+1), donde (Uo, Vo) = (1,0), (U1, V1)= (0, 1),

Congruencia: x mod n ≅ y mod n

MCD Binario

MCD Lehmer

Algoritmo:

1- INPUT: two positive integers x and y in radix b representation, with x >= y.

2- OUTPUT: gcd(x; y).

1. While y >= b do the following:

    1.1 Set x, y, respectively (y^ could be 0).

    1.2 A=1, B=0, C=0,=D=1.

    1.3 While (y + C) != 0 and (y + D) != 0 do the following:

        q1=(x + A)/(y + C), q2=(x + B)/(y + D)

        If q1 != q2 then go to step 1.4.

        t=A - qC, A=C, C=t, t=B - qD, B=D, D=t.

        t=x - qy, x=y, y=t.

    1.4 If B = 0, then t=x mod y, x=y, y=t ; otherwise, t= Ax + By, u=Cx + Dy, x=t , y=u.

2. Compute v = gcd(x; y) using Euclid's Algorithm:

    input x,y where x > y

    while b != 0 do the following:

    r = a mod b , a=b , b=r

3. Return(v)

X	Y	q1	q2	A	B	C	D
768454923	542167814	1	1	1	0	0	1
542167814	226287109	2	2	0	1	1	-1
226287109	89593596	2	2	1	-1	-2	3
89593596	47099917	1	1	-2	3	5	-7
47099917	42493679	1	1	5	-7	-7	10
42493679	4606238	9	9	-7	10	12	-17
4606238	1037537	4	4	12	-17	-115	163
1037537	456090	2	2	-115	163	472	-669
456090	125357	3	3	472	-669	-1059	1501
125357	80019	1	1	-1059	1501	3649	-5172
80019	45338	2	1	3649	-5172	-4708	6673

Comparación

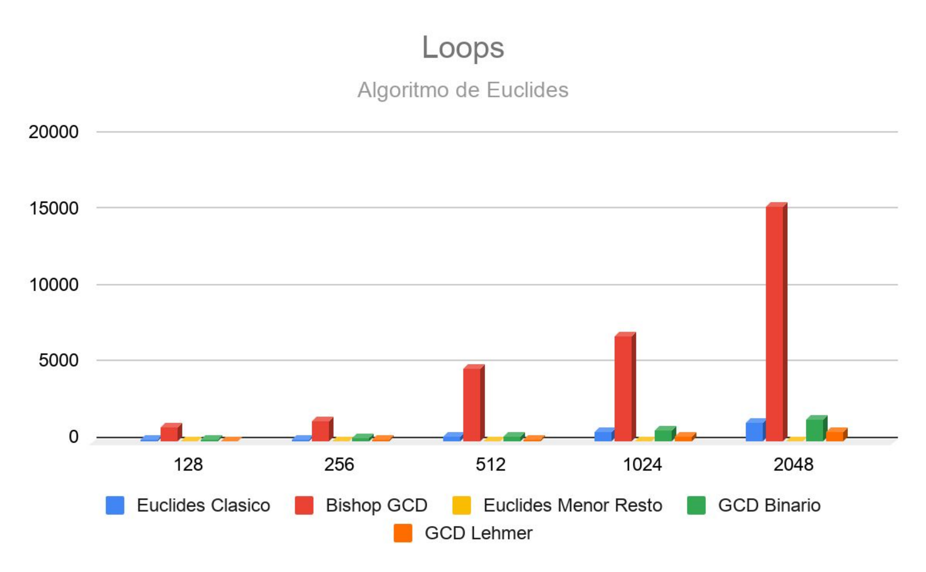
- Tiempo de Compilación:

Bits	Euclides Clásico	Bishop GCD	Euclides Menor Resto	GCD Binario	GCD Lehmer
128	0,384	5,349	0,250	0,300	0,129
256	1,257	9,317	0,000	1,110	0,489
512	5,080	79,031	0,000	3,870	1,796
1024	18,732	242,531	0,000	16,250	6,452
2048	81,118	955,700	0,000	65,420	27,226

- Cantidad de Loops:

Bits	Euclides Clásico	Bishop GCD	Euclides Menor Resto	GCD Binario	GCD Lehmer
128	79	965	50	81	38
256	140	1362	0	180	74
512	309	4814	0	356	157
1024	600	6898	0	753	292
2048	1181	15383	0	1458	594

Comparación Gráfica:



Algoritmo de Euclides Extendido

Binario Extendido

Euclides Extendido

Euclides Extendido Recursivo

El Mejor Euclides Extendido

Euclides Extendido

Teorema de la división:

Sean a, b, q,r $\in$ Z tales que $a = bq + r$ con $b > 0$ y $0 \leq r < b$. Entonces $mcd(a, b) = mcd(b,r)$.

Identidades de Bezout:

Si a, b son dos enteros no ambos cero, existen s,t $\in$ Z (posiblemente no únicos) tales que

$$a(s) + b(t) = mcd(a, b)$$

Algoritmo:

INPUT: two non-negative integers a and b with $a \geq b$.

OUTPUT: $d = \gcd(a, b)$ and integers x, y satisfying $ax + by = d$.

1. If $b = 0$ then set $d \leftarrow a$, $x \leftarrow 1$, $y \leftarrow 0$, and return(d,x,y).
2. Set $x2 \leftarrow 1$, $x1 \leftarrow 0$, $y2 \leftarrow 0$, $y1 \leftarrow 1$.
3. While $b > 0$ do the following:
   3.1 $q \leftarrow \lfloor a/b \rfloor$, $r \leftarrow a - qb$, $x \leftarrow x2 - qx1$, $y \leftarrow y2 - qy1$.
   3.2 $a \leftarrow b$, $b \leftarrow r$, $x2 \leftarrow x1$, $x1 \leftarrow x$, $y2 \leftarrow y1$, and $y1 \leftarrow y$.
4. Set $d \leftarrow a$, $x \leftarrow x2$, $y \leftarrow y2$, and return(d,x,y).

Este algoritmo tiene un tiempo de ejecución de $O(\log(n)^2)$ operaciones de bit.

q	r	x	y	a	b	x_2	x_1	y_2	y_1
—	—	—	—	4864	3458	1	0	0	1
1	1406	1	−1	3458	1406	0	1	1	−1
2	646	−2	3	1406	646	1	−2	−1	3
2	114	5	−7	646	114	−2	5	3	−7
5	76	−27	38	114	76	5	−27	−7	38
1	38	32	−45	76	38	−27	32	38	−45
2	0	−91	128	38	0	32	−91	−45	128

while b > 0 hacer:
q←|_a/b_|, r←a − qb, x←x2 − qx1, y←y2 − qy1.
q=4864/3458=1 r=4864-(1)(3458)=1406 x=1-1(0)=1 y=0-1(1)=-1
iterar hasta que b ya no sea mayor 0

Set d←a, x←x2, y←y2, and return(d,x,y).
d = 38, x = 32, y=-45 ;
4864(32)+3458(-45)=38

Comparación

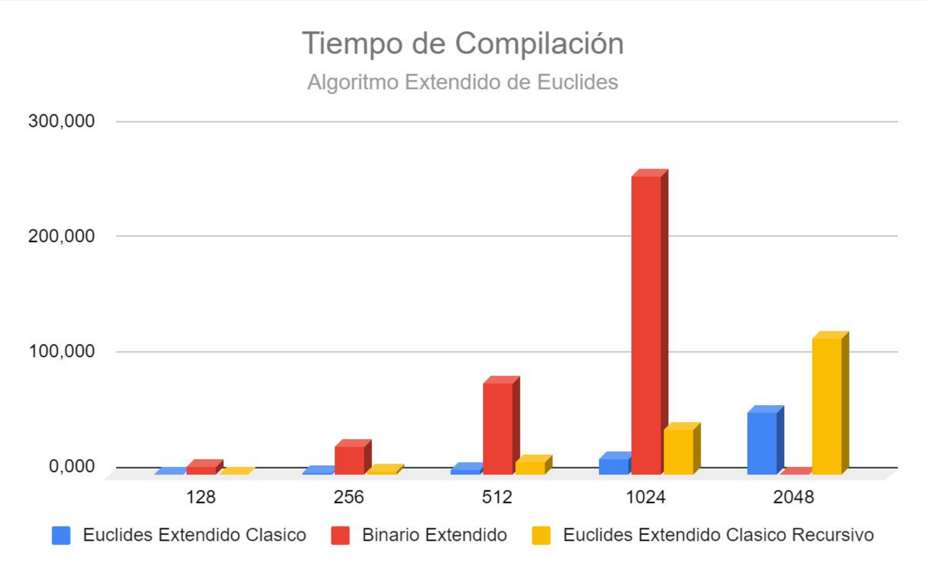
- Tiempo de Compilación:

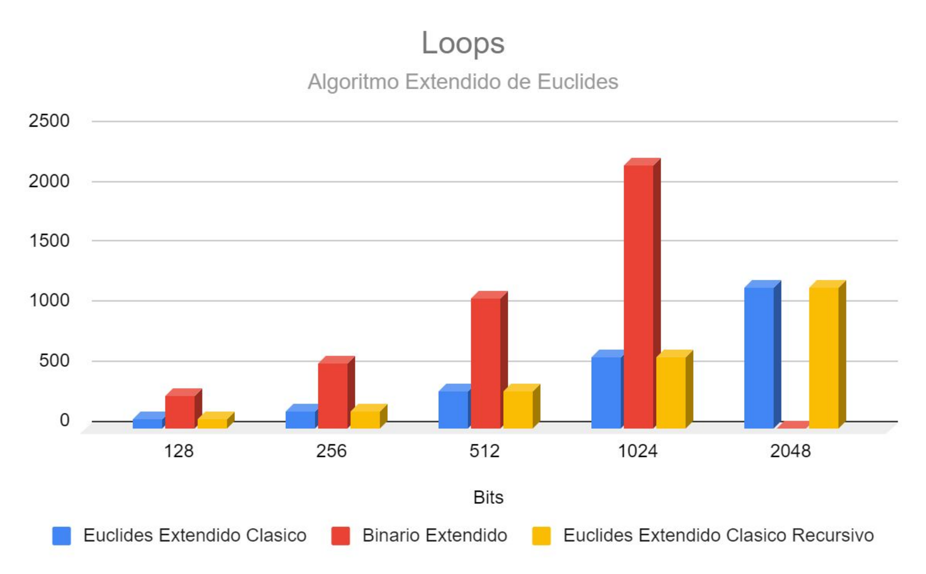
Bits	Euclides Extendido Clásico	Binario Extendido	Euclides Extendido Clásico Recursivo
128	0,278	6,660	0,775
256	0,818	23,681	2,580
512	3,468	79,889	10,469
1024	13,187	258,723	38,531
2048	53,548	0,000	118,001

- Cantidad de Loops:

Bits	Euclides Extendido Clásico	Binario Extendido	Euclides Extendido Clásico Recursivo
128	79	277	79
256	140	545	140
512	309	1079	309
1024	600	2194	600
2048	1181	0	1181

Comparación Gráfica:





Conclusiones:

- El Algoritmo de Lehmer es el más eficiente.Siendo el Binario el segundo más eficiente.
- Buen uso de algoritmos óptimos a través de simplificaciones.
- El Algoritmo Extendido de Euclides es más eficiente que su mismo recursivo y binario extendido.