Departamento de Matemática y Ciencia de la Computación

Licenciatura en Ciencia de la Computación

Laboratorio N°2

Máximo Común Divisor

Matemática Computacional
Profesor: Nicolas Thériault

Matías Daza A.
María Maldonado P.

Índice general

1. Introducción 2
2. Algoritmo 1: Algoritmo Euclideano Simple 2
   2.1. Formulación del experimento 2
   2.2. Gráfico de desempeño 2
3. Algoritmo 2: Algoritmo Euclideano 3
   3.1. Formulación del experimento 3
   3.2. Gráfico de desempeño 3
4. Algoritmo 3: Algoritmo Euclideano Binario 5
   4.1. Formulación del experimento 5
   4.2. Gráfico de desempeño 5
5. Algoritmo 4: Algoritmo Euclideano Extendido 7
   5.1. Formulación del experimento 7
   5.2. Gráfico de desempeño 7
6. Algoritmo 5: Algoritmo Euclideano Extendido Binario 9
   6.1. Formulación del experimento 9
   6.2. Gráfico de desempeño 9
7. Conclusiones 11
8. Estructuras de Datos Utilizadas 11
9. Lectura de programa desde la consola 11
10. Implementación 11
11. Plataforma Computacional 11
    11.1. Librerías 11
12. Bibliografía 11

1. Introducción

El siguiente informe tiene como objetivo la implementación de algoritmos que resuelven el problema de determinar el máximo común divisor de dos enteros positivos a y b utilizando 5 estrategias diferentes para en las cuales se calculan las iteraciones y costo computacional de cada estrategia.

2. Algoritmo 1: Algoritmo Euclideano Simple

El algoritmo Euclideano es uno de los primeros algoritmos conocidos. En su versión extendida, tiene aplicaciones muy importantes en teoría de números, álgebra, teoría de códigos, criptografía,etc. En su formato original (como descrita por Euclides alrededor del año 300 a.C.), el algoritmo Euclideano obtiene el máximo común divisor de dos enteros de manera inductiva haciendo solamente diferencias entre dos enteros positivos.

2.1. Formulación del experimento

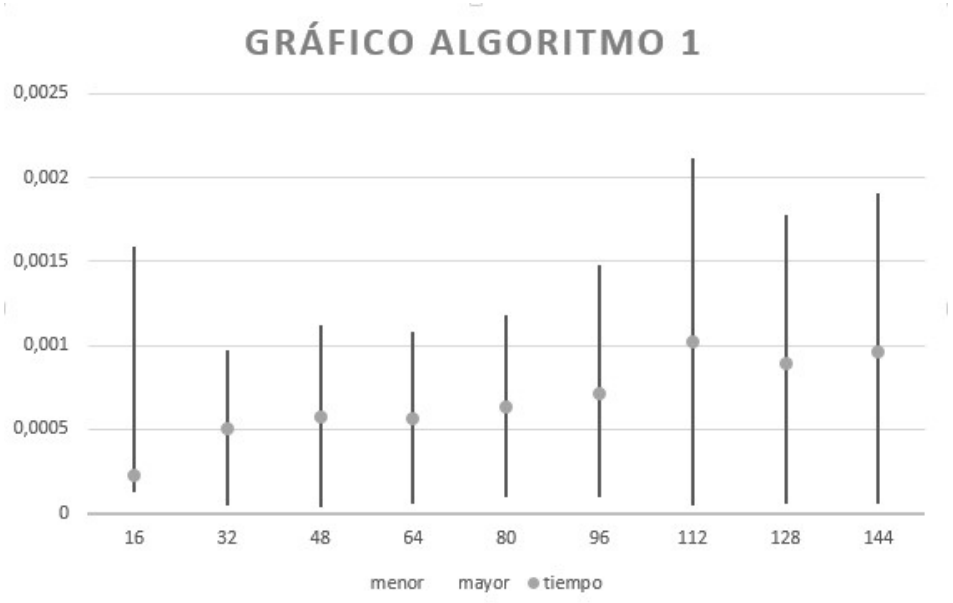
Experimento con a y b dados y tiempo de ejecución.

a	b	gcd(a,b)	Vueltas	Tiempo de ejecución (s)
291	252	3	19	0.000299
85652	16261	161	24	0.000161
914407221	897279761	3	237	0.000187
16534528044	8332745927	43	129	0.000215

2.2. Gráfico de desempeño

Compare los tiempos del **Algoritmo Euclideano Simple** con varias combinaciones de entradas: a lo menos 20 pares (a,b) para cada tamaño de a (i.e. $log_2(a)$), considerando a lo menos 10 tamaños diferentes.

Observación: Archivo con las distintas entradas está incluido en el directorio principal.



3. Algoritmo 2: Algoritmo Euclideano

El mayor problema con el método 1 ocurre cuando a es mucho más grande que b, porque el algoritmo dará muchas vueltas repetitivas restando b una y otra vez (en general, a/b pasos), por ejemplo si a = 23,141,633,522 y b = 17. Como sabemos que el algoritmo repetirá la misma sustracción hasta que la resta sea menor que b (paso 4 del Algoritmo 1), una mejora muy natural consiste en utilizar el resultado de la división Euclideana (división de enteros con resta), es decir se escribe a = qb + r con r, q ∈ Z y $0 \leq r < b$. En la notación moderna, escribimos r = a MOD b para indicar la resta de esta división.

3.1. Formulación del experimento

Experimento con a y b dados y tiempo de ejecución.

a	b	gcd(a,b)	Vueltas	Tiempo de ejecución (s)
291	252	3	8	0.000201
85652	16261	161	12	0.000174
914407221	897279761	7	30	0.000292
16534528044	8332745927	43	44	0.000162

a=43312793054108111		
b=43263441545690516		
gcd(a,b)	Vueltas	Tiempo de ejecución (s)
1	58	0.000288

a=627710173538668076383578942320766641608390870039032496127 9		
b=233567642346898765322334567888765432345 67		
gcd(a,b)	Vueltas	Tiempo de ejecución (s)
1	166	0.000345

a=6818536214948665048512398719751378771718765228936247143232679847194284772046122091468887714		
b=2123010620889223608977186369097185643295866022044800274887840195619371824911497555030419 50		
gcd(a,b)	Vueltas	Tiempo de ejecución (s)
2	332	0.000383

3.2. Gráfico de desempeño

Compare los tiempos del **Algoritmo Euclideano** con varias combinaciones de entradas: a lo menos 20 pares (a,b) para cada tamaño de a (i.e. $log_2(a)$), considerando a lo menos 10 tamaños diferentes.

Observación: Archivo con las distintas entradas está incluido en el directorio principal.

	menor	mayor	Tiempo de ejecución (s)
16	0.000106	0.001175	0.00073211
32	0.000024	0.000826	0.00042168
48	0.000026	0.001192	0.00051842
64	0.000035	0.001062	0.00054821
80	0.000036	0.001165	0.00058016
96	0.000034	0.001073	0.00059868
112	0.000031	0.000931	0.00048342
128	0.000031	0.001036	0.00053563
144	0.000037	0.001103	0.00054589

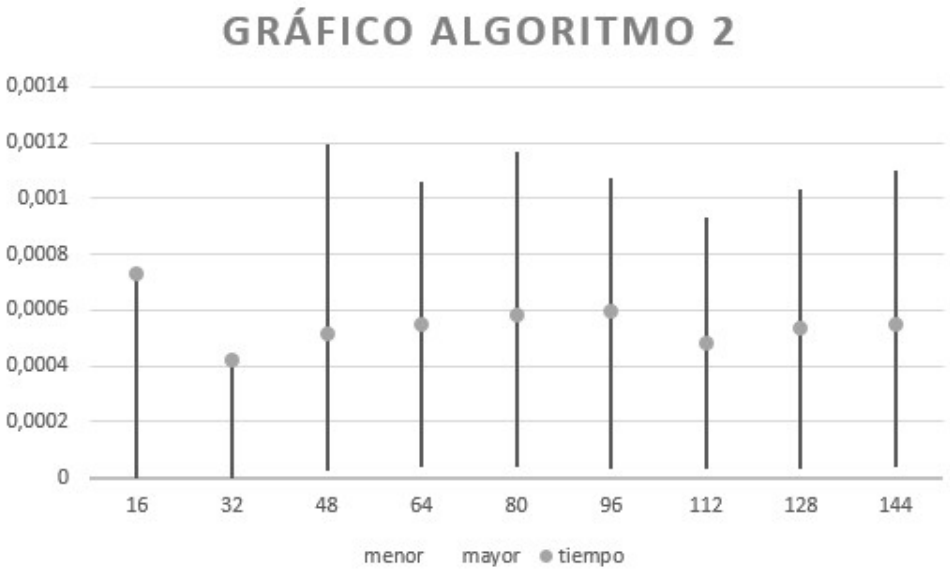


GRÁFICO ALGORITMO 2
0,0014
0,0012
0,001
0,0008
0,0006
0,0004
0,0002
0
16
32
48
64
80
96
112
128
144
menor
mayor
tiempo

4. Algoritmo 3: Algoritmo Euclideano Binario

Con la llegada de los computadores y el desarrollo de la teoría de la complejidad, se observó que utilizar la división Euclideana (división de enteros con resta) no siempre es lo más eficiente dado que es una operación relativamente más "costosa" que la sustracción. En este caso, se puede volver a un algoritmo más cercano a la descripción original, pero tomando ventaja que la división por 2 es muy sencilla para un computador (debido que trabaja en base 2 en vez de base 10). Así apareció una versión llamada "algoritmo Euclideano binario".

4.1. Formulación del experimento

Experimento con a y b dados y tiempo de ejecución.

a	b	gcd(a,b)	Vueltas	Tiempo de ejecución (s)
291	252	3	10	0.000194
85652	16261	161	9	0.000230
914407221	897279761	7	27	0.000223
16534528044	8332745927	43	40	0.000319

a=43312793054108111		
b=43263441545690516		
gcd(a,b)	Vueltas	Tiempo de ejecución (s)
1	53	0.000320

a=627710173538668076383578942320766641608390870039032496127		
b=233567642346898765322334567888765432345 67		
gcd(a,b)	Vueltas	Tiempo de ejecución (s)
1	179	0.000350

a=6818536214948665048512398719751378771718765228936247143232679847194284772046122091468887714		
b=212301062088922360897718636909718564329586602204480027488784019561937182491149755503041950		
gcd(a,b)	Vueltas	Tiempo de ejecución (s)
2	314	0.000485

4.2. Gráfico de desempeño

Compare los tiempos del **Algoritmo Euclideano Binario** con varias combinaciones de entradas: a lo menos 20 pares (a,b) para cada tamaño de a (i.e. $log_2(a)$), considerando a lo menos 10 tamaños diferentes.

Observación: Archivo con las distintas entradas está incluido en el directorio principal.

	menor	mayor	Tiempo de ejecución (s)
16	0.000070	0.000946	0.00051153
32	0.000014	0.000530	0.00027389
48	0.000017	0.000590	0.00030432
64	0.000019	0.000636	0.00032679
80	0.000024	0.000883	0.00043789
96	0.000028	0.000817	0.00042605
112	0.000031	0.000849	0.00044068
128	0.000034	0.000965	0.00049447
144	0.000039	0.000989	0.00051389


GRÁFICO ALGORITMO 3
0,0012
0,001
0,0008
0,0006
0,0004
0,0002
0
16
32
48
64
80
96
112
128
144
menor
mayor
tiempo

5. Algoritmo 4: Algoritmo Euclideano Extendido

Aunque encontrar el máximo común divisor (de dos enteros, polinomios, etc) tiene varias aplicaciones, el algoritmo Euclideano alcanza su máximo potencial en su forma extendido, cuando calcula también dos enteros s y t tales que:

$$d = gcd(a, b) = s \cdot a + t \cdot b$$

Para obtener s y t, el algoritmo Euclideano extendido guarda una cuenta de como cada a y b intermedio puede ser escrito a partir de a y b iniciales (que podemos denotar a 0 y b 0).

5.1. Formulación del experimento

Experimento con a y b dados y tiempo de ejecución.

a	b	gcd(a,b)	Tiempo de ejecución (s)
291	252	3	0.000145
85652	16261	161	0.000144
914407221	897279761	7	0.000105
16534528044	8332745927	43	0.000209

a=43312793054108111	
b=43263441545690516	
gcd(a,b)	Tiempo de ejecución (s)
1	0.000161

a=627710173538668076383578942320766641608390870039032496127 9	
b=233567642346898765322334567888765432345 67	
gcd(a,b)	Tiempo de ejecución (s)
1	0.000239

a=6818536214948665048512398719751378771718765228936247143232679847194284772046122091468887714	
b=2123010620889223608977186369097185643295866022044800274887840195619371824911497555030419 50	
gcd(a,b)	Tiempo de ejecución (s)
2	0.000216

5.2. Gráfico de desempeño

Compare los tiempos del **Algoritmo Euclideano Extendido** con varias combinaciones de entradas: a lo menos 20 pares (a,b) para cada tamaño de a (i.e. $log_2(a)$), considerando a lo menos 10 tamaños diferentes.

Observación: Archivo con las distintas entradas está incluido en el directorio principal.

	menor	mayor	Tiempo de ejecución (s)
16	0.000172	0.001744	0.00099853
32	0.000052	0.001626	0.00084237
48	0.000064	0.001763	0.00092016
64	0.000069	0.001728	0.00091408
80	0.000070	0.002020	0.00101505
96	0.000081	0.002052	0.00107289
112	0.000098	0.002032	0.00109184
128	0.000077	0.002198	0.00111058
144	0.000108	0.002283	0.00121321


GRÁFICO ALGORITMO 4
0,0025
0,002
0,0015
0,001
0,0005
0
16
32
48
64
80
96
112
128
144
menor
mayor
tiempo

6. Algoritmo 5: Algoritmo Euclideano Extendido Binario

6.1. Formulación del experimento

Experimento con a y b dados y tiempo de ejecución.

a	b	gcd(a,b)	Tiempo de ejecución (s)
291	252	3	0.000195
85652	16261	161	0.000322
914407221	897279761	7	0.000357
16534528044	8332745927	43	0.000332

a=43312793054108111	
b=43263441545690516	
gcd(a,b)	Tiempo de ejecución (s)
1	0.000395

a=627710173538668076383578942320766641608390870039032496127 9	
b=233567642346898765322334567888765432345 67	
gcd(a,b)	Tiempo de ejecución (s)
1	0.000557

a=6818536214948665048512398719751378771718765228936247143232679847194284772046122091468887714	
b=2123010620889223608977186369097185643295866022044800274887840195619371824911497555030419 50	
gcd(a,b)	Tiempo de ejecución (s)
2	0.000696

6.2. Gráfico de desempeño

Compare los tiempos del **Algoritmo Euclideano Extendido Binario** con varias combinaciones de entradas: a lo menos 20 pares (a,b) para cada tamaño de a (i.e. $log_2(a)$), considerando a lo menos 10 tamaños diferentes.

Observación: Archivo con las distintas entradas está incluido en el directorio principal.

	menor	mayor	Tiempo de ejecución (s)
16	0.000092	0.001098	0.00061042
32	0.000034	0.001229	0.00065784
48	0.000040	0.001135	0.00058526
64	0.000047	0.001358	0.00070553
80	0.000062	0.001704	0.00087679
96	0.000071	0.001871	0.00093347
112	0.000082	0.002048	0.00106258
128	0.000092	0.002366	0.00122768
144	0.000112	0.002366	0.00126637


GRÁFICO ALGORITMO 5
0,0025
0,002
0,0015
0,001
0,0005
0
16
32
48
64
80
96
112
128
144
menor
mayor
tiempo

7. Conclusiones

En el algoritmo euclidiano binario, se cambian las divisiones por restas, aunque el tiempo aumenta no es tan elevado como en el algoritmo eucliadiano simple, ya que al hacer restas y no divisiones se vuelve un algoritmo más eficiente con aquellos números que sean exactamente divisibles por 2, tornando mayor complejidad para los valores s y t que no sean divisibles por 2.

Por otro lado, al utilizar el algoritmo euclidiano extendido binario, se suma al costo la obtención de los valores s y t, para aquellos valores a y b que sean pares.

8. Estructuras de Datos Utilizadas

Se utilizaron variables GMP las cuales trabajan con punteros.

9. Lectura de programa desde la consola

La lectura del programa se indica a continuación:

Para los 5 programas i = 1,2,3,4,5.

Para compilar:

- gcc -o Ejecutari lab2-i.c -lgmp

Para ejecutar:

- ./Ejecutari a b

10. Implementación

El algoritmo está implementado en lenguaje C.

11. Plataforma Computacional

El programa fue ejecutado en un computador con las siguientes características:

- **Procesador:** Intel Core i3-350M CPU 2.27GHz
- **Memoria RAM:** 5,6 GiB
- **Tipo de SO:** 64 bits
- **Sistema Operativo:** Linux - Ubuntu 14.04 LTS

11.1. Librerías

- gmp.h
- stdio.h
- time.h

12. Bibliografía

- Librería GMP (http://gmplib.org).