Universidad Católica
San Pablo

CIENCIA DE LA COMPUTACION

ALGORITMOS DE MAXIMO COMUN DIVISOR (EUCLIDES)

ALGEBRA ABSTRACTA

ANDRES CUSIRRAMOS

MARCELO BUSTIOS

TERCER SEMESTRE
2019

"El alumno declara haber realizado el presente trabajo de acuerdo a las normas de la Universidad Católica San Pablo"

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _-

Informe Algoritmos de Maximo Común Divisor de Euclides

En este informe se va detallar la lógica de cada uno de los algoritmos entregados en clase, los tiempos de ejecución, cantidad de vueltas realizadas y su eficiencia.

Algoritmo 1:

En este algoritmo se da un bucle while en el cual se realiza una operación modulo y se intercambian los valores.

-Pseudo Código:

P1 Leer a y b

P2 r = resto de dividir a entre b

P3 si r = 0 entonces la función(a,b) = b. FIN

P4 si no a = b, b = r

P5 ir al P2

-Código:

```
unsigned long long Algoritmo1(unsigned long long  a, unsigned long long  b) {
    while (true) {

        unsigned long long r = mod(a,b);
        if (r == 0) {
            return b;
        } else {
            a = b;
            b = r;
        }

    }
}
```

-Ejemplo:

a=412 ,b=260

V1: n= a%b=152
n≠0
a=260
b=152

V2: n=a%b=108
n≠0
a=162
b=108

V3: n= a%b=44
n≠0
a=108
b=44

V4: n=a%b=20
n≠0
a=44
b=20

V5: n=a%b=4
n≠0
a=20
b=4

V6: n=a%b=0
n=0
Termina y retorna 4

-Por lo tanto el resultado es 4 y se hizo en 6 pasos.

ALGORITMO 1	5 CIFRAS	10 CIFRAS	15 CIFRAS	20 CIFRAS
TIEMPO	0,000 02			
#N DE LOOPS	5			
IMPAR,PAR	0,000 44	0,000 52	0,000 51	0,000 55
#N DE LOOPS	7	22	21	17
PAR,IMPAR	0,000 43	0,000 52	0,000 52	0,000 55
#N DE LOOPS	9	19	22	18
PAR,PAR	0,000 43	0,000 52	0,000 53	0,000 57
#N DE LOOPS	11	17	16	17
IMPAR,IMPAR	0,000 43	0,000 52	0,000 53	0,000 59
#N DE LOOPS	13	14	21	19

-En el código se puede observar que es eficiente con números cortos y largos
-En la tabla puede ver la cantidad de loops y el tiempo, tiene una cantidad de vueltas considerable y en tiempo es muy parecido en humero de 5 o de 20 no varia demasiado.Es un código eficiente para numero cortos y largos. Los primeros valor de la columna 2 representa el tiempo y los loops del ejemplo.

Algoritmo 2:

En este Algoritmo se usa un bucle while y dos condiciones en las cuales se realiza operaciones matematicas simples y intercambio de variables.

-Pseudo Código:

P1 Leer a y b

P2 r = resto de dividir a entre b

P3 si r = 0 entonces la función(a,b) = b. FIN

P4 Si r > b/2 entonces r = b-r

P5 a = b, b = r

P6 ir al P2

-Código:

```
unsigned long long Algoritmo2(unsigned long long a ,unsigned long long b){
    while(true){
        unsigned long long resta=mod(a,b);
        if(resta==0){return b;}
        if(resta>b/2){resta=b-resta;}
        a=b;
        b=resta;
    }
}
```

-Ejemplo:

a=412 ,b=260

V1: n=a%b=152
n≠0
n>b/2=108
a=260
b=108

V3: n=a%b=20
n≠0
n>b/2 no entra
a=44
b=20

V5: n=a%b=0
n≠0
Termina y retorna 4

V2: n=a%b=44
n≠0
n>b/2 no entra
a=108
b=44

V4: n=a%b=4
n≠0
n> b/2 no entra
a=20
b=4

-Por lo tanto retorno 4 y se termina en 5 pasos.

ALGORITMO 2	5 CIFRAS	10 CIFRAS	15 CIFRAS	20 CIFRAS
TIEMPO	0,000 45			
#N DE LOOPS	6			
IMPAR,PAR	0,000 57	0,000 53	0,000 68	0,000 89
#N DE LOOPS	10	8	14	13
PAR,IMPAR	0,000 74	0,000 67	0,000 76	0,000 68
#N DE LOOPS	7	10	13	11
PAR,PAR	0,000 48	0,000 74	0,000 61	0,000 65
#N DE LOOPS	4	12	11	11
IMPAR,IMPAR	0,000 52	0.000 68	0,000 66	0,000 56
#N DE LOOPS	6	13	14	9

-En la tabla se ve que termina el código y da el se puede observar que da menos vueltas que el primero pero en un tiempo ligeramente mayor pero a pesar de tener números mas grandes el numero de vueltas se ve afecta muy poco. Los primeros valor de la columna 2 representa el tiempo y los loops del ejemplo.

Algoritmo 3:

En este algoritmo se puede ver que es recursivo y que solo tiene un condicional if en cada el segundo valor sea cero en caso de no serlo los parámetros de la función cambian y se pone al segundo valor como primero y luego se saca modulo de los dos valores como segundo valor.

-Pseudo Código:

```
Euclidesmcd(a,b)
    Entrada: enteros no negativos a, b
    Salida: mcd(a,b)
    If (b==0)
        Return a
    Return Euclidesmcd(b,a mod b)
```

-Código:

```
unsigned long long Algoritmo3(unsigned long long a ,unsigned long long b){
    if(b==0){return a;}
    return Algoritmo3(b,mod(a,b));
}
```

-Ejemplo:

a=412 ,b=260

V1: b≠0
Algoritmo3(260,a%b)

V2: b≠0
Algoritmo3(152,108)

V3: b≠0
Algoritmo3(108,74)

V4: b≠0
Algoritmo3(44,20)

V5: b≠0
Algoritmo3(20,4)

V6 b=0
Termina y retorna 4

-Por lo tanto retorna 4 y termina en 6 pasos.

ALGORITMO 3	5 CIFRAS	10 CIFRAS	15 CIFRAS	20 CIFRAS
TIEMPO	0,000 52			
#N DE LOOPS	6			
IMPAR,PAR	0,000 54	0,000 78	0,000 76	0,000 77
#N DE LOOPS	7	22	23	21
PAR,IMPAR	0,000 55	0,000 71	0,000 68	0,000 68
#N DE LOOPS	7	19	18	19
PAR,PAR	0,000 53	0,000 77	0,000 66	0,000 71
#N DE LOOPS	5	22	12	19
IMPAR,IMPAR	0,000 55	0.000 72	0,000 74	0,000 86
#N DE LOOPS	7	20	19	24

-En la tabla se puede ver con números de cinco cifras el tiempo y la cantidad de vueltas es mucho menor que aun numero con 10 dígitos pero con números de mas de diez dígitos el tiempo y las vueltas no tiene mucha variación, pero si ve una gran diferencia entre lo números de 5 y 10 y dígitos a mas. Los primeros valor de la columna 2 representa el tiempo y los loops del ejemplo.

Algoritmo 4:

En este código se puede ver que una secuencia de condicionales en los cuales se va comprobando si los números son pares o impares o las dos condiciones al mismo tiempo, condiciones en las cuales se aplica valor absoluto y operaciones de modulo.Es un algoritmo de carácter recursivo.

-Pseudo Código:

Algorithm Binary-Gcd.
Input: $a, b \in \mathbb{Z}$.
Output: $\gcd(a, b)$.

1. If $|b| > |a|$, return Binary-Gcd(b, a).
2. If $b = 0$, return a.
3. If a and b are both even then return $2 \cdot$ Binary-Gcd$(a/2, b/2)$.
4. If a is even and b is odd then return Binary-Gcd$(a/2, b)$.
5. If a is odd and b is even then return Binary-Gcd$(a, b/2)$.
6. Otherwise return Binary-Gcd$((|a| - |b|)/2, b)$.

-

Código:

```
unsigned long long  Algoritmo4(unsigned long long  a, unsigned long long b){
    //1
    if (abso(b)>abso(a)){
        return Algoritmo4(b, a);
    }
    //2
    if(b==0){
        return a;
    }
    //3
    if ((mod(a, y: 2)==0) and (mod(b, y: 2)==0)){
        return 2*Algoritmo4( a: a/2,  b: b/2);
    }
    //4
    if((mod(a, y: 2)==0)and(mod(b, y: 2)!=0)){
        return Algoritmo4( a: a/2, b);
    }
    //5
    if((mod(a, y: 2)!=0)and(mod(b, y: 2)==0)){
        return Algoritmo4(a, b: b/2);}

    return Algoritmo4(((abso(a)-abso(b))/2),b );


}
```

-Ejemplo:

a=412 ,b=260

V1: 412 es par y 260 es par
Algoritmo4(206,130)
206 es par y 130 es par
Algoritmo4(206,130)
103 es impar y 65 es impar
Algoritmo4((I103I-I65I)/2,65)

V2: 19 es impar y 65 es impar
I19I >I65I
Algoritmo4(65,19)

V3: 23 es impar y 19 es impar
Algoritmo4((I23I-I19I)/2,19)

V4: 2 es par y 19 es impa
I2I>I19I
Algoritmo4(19,2)
19 es impar y 2 es par
Algoritmo4(19,1)

V5: 9 es impar y 1 es impar
Algoritmo4((I9I-I1I)/2,1)
4 es par y 1 es impar

V6: Algoritmo4(4,1)
0,1
Termina y retorna 4

-Por lo tanto retorna 4 y termina en 6 pasos.

ALGORITMO 4	5 CIFRAS	10 CIFRAS	15 CIFRAS	20 CIFRAS
TIEMPO	0,000 58			
#N DE LOOPS	6			
IMPAR,PAR	0,000 78	0,000 88	0,000 87	0,000 89
#N DE LOOPS	13	22	20	21
PAR,IMPAR	0,000 71	0,000 92	0,000 88	0,000 92
#N DE LOOPS	11	24	23	23
PAR,PAR	0,000 71	0,000 89	0,000 88	0,000 92
#N DE LOOPS	10	23	21	23
IMPAR,IMPAR	0,000 71	0.000 89	0,000 88	0,000 94
#N DE LOOPS	10	23	21	25

-En la tabla se puede observar que las vueltas con números de 5 dígitos no son muy altas pero al ingresar números de 10 dígitos se duplican la cantidad de vueltas pero al ingresar de 15 o de 20 se mantiene casi la misma cantidad de estas, en tiempo es igual no varia en relación con los números 10,15,y 20 dígitos pero con los de 5 son considerablemente mucho mas bajos. Los primeros valor de la columna 2 representa el tiempo y los loops del ejemplo.

Algoritmo 5:

Todo el código se ejecuta en base a una condicional, esta compuesto por dos whiles uno en el cual se verifica si los valores son pares o no y luego se opera las variables y una variable auxiliaren el segundo while se tienen otros dos mas dentro de este donde según las condiciones se operan las variables, una ultima conditional donde se remplazan valores.

-Pseudo Código:

Algorithm Binary gcd algorithm

INPUT: two positive integers x and y with $x \geq y$.
OUTPUT: $\gcd(x, y)$.

1. $g \leftarrow 1$.
2. While both x and y are even do the following: $x \leftarrow x/2$, $y \leftarrow y/2$, $g \leftarrow 2g$.
3. While $x \neq 0$ do the following:
   3.1 While x is even do: $x \leftarrow x/2$.
   3.2 While y is even do: $y \leftarrow y/2$.
   3.3 $t \leftarrow |x - y|/2$.
   3.4 If $x \geq y$ then $x \leftarrow t$; otherwise, $y \leftarrow t$.
4. Return$(g \cdot y)$.

Código:

```
unsigned long long Algoritmo5( unsigned long long  a, unsigned long long b){
    int g=1;
    if(a>=b){
        while((mod(a, y: 2)==0)and(mod(b, y: 2)==0) ){
            a=a/2;
            b=b/2;
            g=g*2;
        }
        while(a!=0){
            while(mod(a, y: 2)==0){
                a=a/2;
            }
            while(mod(b, y: 2)==0){
                b=b/2;
            }
            int t=(abso( a: a-b))/2;
            if(a>=b){
                a=t;
            }
            else{
                b=t;
            }
        }
    }
    return (g*b);
}
```

Ejemplo:

a=412 ,b=260

V1: 412>=260
412 y 260 son pares
g=2
103 y 65 son impares

V2: T=((I103-65I)/2)
103 >=65
a=t
g=4
19,65

V3: 19 y 65 son impares
T=((I19-65I)/2)
19>=65
b=t
19,23

V4: 19 y 23 son impares
T=((I19-23I)/2)
19>=23
b=t
19,2

V5: 19 es impar y 2 es par
2/2=1
19,1
19 y 1 son impares
T=((I19-1)/2)
19>=1
a=t
9,3

V6: 9 y 3 son impares
T=((I9-3I)/2)
9>=3
a=t
4,1

V7: 2 es impar y 1 es impar
2/2=1
1,1
1y 1 son impares
T=((I1-1I)/2)
1>=1
a=t
0,1
Ya no entra al if
g=4 y b=1
Retorna (g*b)=4

-Por lo tanto retorna 4 y termina en 7 pasos.

ALGORITMO 5	5 CIFRAS	10 CIFRAS	15 CIFRAS	20 CIFRAS
TIEMPO	0,000 49			
#N DE LOOPS	7			
IMPAR,PAR	0,000 53	0,000 54	0,000 54	0,000 55
#N DE LOOPS	11	26	29	29
PAR,IMPAR	0,000 51	0,000 49	0,000 56	0,000 57
#N DE LOOPS	12	27	27	27
PAR,PAR	0,000 67	0,000 59	0,000 60	0,000 58
#N DE LOOPS	14	21	28	25
IMPAR,IMPAR	0,000 53	0.000 63	0,000 59	0,000 60
#N DE LOOPS	16	25	30	28

-Se puede observar en la tabla que los tiempos no varian muchos a pesar de la cantidad de dígitos que se ingresen los que si se ven afectados son la cantidad de vueltas siendo mayor que los demás algoritmos mostrados en este informe se puede determinar que la cantidad de bucles haven a este código menos eficiente en cuestión de tiempo y en bucles con respecto a los demás . Los primeros valor de la columna 2 representa el tiempo y los loops del ejemplo.

Algoritmo 6:

En este algoritmo se puede observar que su estructura principal parte de un bucle while donde se comprueba que los valores sean diferentes uno de otro luego pones ver una estructura condicional en la cual se da que el primer valor debe ser mayor al segundo sino se operan los resultados y luego retorna el primer valor.

-Pseudo Código:

```
GCD(a, b)
    begin
        while a ≠ b do
            if a > b then
                a=  a – b
            else
                b = b – a
        return a
    end
```

-Código:

```
unsigned long long Algoritmo6(unsigned long long a, unsigned long long b){
    while(a!=b){
        if(a>b){
            a=a-b;
        }
        else{
            b=b-a;
        }
    }
    return a;
}
```

Ejemplo:

a=412 ,b=260

```
152 , 260
152 , 108
44 , 108
44 , 64
44 , 20
24 , 20
4 , 20
4 , 16
4 , 12
4 , 8
4 , 4
4
```

En este Algoritmo se comprueba que no sean iguales si no son iguales pasan a ver si a>b, si este es así se hace a=a-b, caso contrario se b=b-a y así hasta que se inculpa la condición principal que seria el que no sean igual, al incumplirse se imprime a.

-Por lo tanto retorna 4 y termina en 11 pasos.

ALGORITMO 6	5 CIFRAS	10 CIFRAS	15 CIFRAS	20 CIFRAS
TIEMPO	0,000 45			
#N DE LOOPS	11			
IMPAR,PAR	0,000 52	0,000 46	0,000 45	0,000 46
#N DE LOOPS	30	66	201	569
PAR,IMPAR	0,000 48	0,000 63	0,000 48	0,000 59
#N DE LOOPS	28	63	118	358
PAR,PAR	0,000 60	0,000 57	0,000 88	0,000 48
#N DE LOOPS	42	59	111	361
IMPAR,IMPAR	0,000 46	0.000 48	0,000 49	0,000 49
#N DE LOOPS	41	97	1635	680

-En esta tabla se puede ver que este el algoritmo con mas numero de loops en ya que se estructura depende una condición que se repite hasta incumplirse se puede ver que el tiempo no es tan alto respecto a los demás algoritmos pero en números de vueltas es muy alto lo cual no lo hace tan eficiente porque se esta recorriendo muchas veces un bucle lo cual afecta la memoria por lo tanto a pesar de ser relativamente rápido no es tan eficiente por la cantidad de loops que posee. Los primeros valor de la columna 2 representa el tiempo y los loops del ejemplo.

Algoritmo 7:

En este algoritmo se puede ver que se le asignan dos nuevos nombres a los valores principales luego se genera un bucle el cual se seguirá cumpliendo hasta que el segundo termino sea menor que cero dentro del while se crean otras dos variables una que divide a los valores y la otra usa el algoritmo de la division para hallar el residuo se igual valores y se retorna el primer valor.

-Pseudo Código:

$r_1 \leftarrow a; \quad r_2 \leftarrow b;$ (Initialization)

while $(r_2 > 0)$

{

$q \leftarrow r_1 / r_2;$

$r \leftarrow r_1 - q \times r_2;$

$r_1 \leftarrow r_2; \quad r_2 \leftarrow r;$

}

$\gcd(a, b) \leftarrow r_1$

```
unsigned long long Algoritmo7(unsigned long long a,unsigned long long b){
    unsigned long long r1=a;
    unsigned long long r2=b;
    while(r2>0){
        unsigned long long q=r1/r2;
        unsigned long long r=r1-q*r2;
        r1=r2;
        r2=r;
    }
    return r1;
}
```

-Código:
Ejemplo:

a=412 ,b=260

V1: 260>0
q=412/260
r=412-q*260
412=260
260=r
(260,152)

V2: 152>0
q=260/152
r=260-q*152
260=152
152=r
(152,108)

V3: 108>0
q=152/108
r=152-q*108
152=108
108=r
(108,44)

V4: 44>0
q=108/44
r=108-q*44
108=44
44=r
(44,20)

V5: 20>0
q=44/20
r=44-q*20
44=20
20=r
(20,4)

V6: 4>0
q=20/4
r=20-q*4
20=4
4=r
(4,0)

V7: 0>0
No entra
Termina y retorna el valor de r1 =4

-Por lo tanto retorna 4 y termina en 7 pasos.

ALGORITMO 7	5 CIFRAS	10 CIFRAS	15 CIFRAS	20 CIFRAS
TIEMPO	0,000 24			
#N DE LOOPS	7			
IMPAR,PAR	0,000 48	0,000 53	0,000 50	0,000 47
#N DE LOOPS	11	20	34	38
PAR,IMPAR	0,000 49	0,000 50	0,000 50	0,000 49
#N DE LOOPS	10	24	35	43
PAR,PAR	0,000 48	0,000 49	0,000 48	0,000 50
#N DE LOOPS	12	23	28	39
IMPAR,IMPAR	0,000 48	0.000 49	0,000 49	0,000 47
#N DE LOOPS	13	26	29	39

-En esta tabla se puede ver como los tiempos con respecto a la cantidad de dígitos es muy parecida ya que no varia en un gran cantidad y la cantidad de loops que da el algoritmo es incrementando de 10 en 10 aproximadamente es un algoritmo no recursivo lo cual lo hace mucho mejor y escaso igual de eficiente con números de 5 dígitos como con números de 20.Los primeros valor de la columna 2 representa el tiempo y los loops del ejemplo.

En conclusion los algoritmos de de recursividad no son tan efecientes y requieren mas tiempo de ejecucion y hemos llegado a la conclusion de algoritmo 4 es el mejor ya que al no ser recursivo y no tener bucles en los cuales son mas costosos.