

CIENCIA DE LA COMPUTACIÒN

Algoritmos para MCD

Algebra Abstracta

Rubén Félix

Guzmán Chiroque

3er SEMESTRE

2019

Algoritmo 1:

Sea a 412 y b 260.

Vemos que se realiza lo siguiente:

a b r

412 260 108 Vuelta 1

260 108 44 Vuelta 2

108 44 20 Vuelta 3

44 20 4 Vuelta 4

20 4 0 Vuelta 5

En total se realizó 5 usos del módulo.

Para realizar el modulo se hiso uso del menor resto.

Siendo este el código:

Int Función\_menor\_resto(int a, int b)

a =|a|

b =|b|

entero r = a-b\*(a/b) // La división es entera, de caso contrario usar floor(a/b+0.5) de la librería math

si b-r es menor a r retornar b-r

retornar r

Donde |x| es una pregunta simple, en total se usó 5 divisiones, 5 multiplicaciones y 15 restas, con el menor resto la cantidad de restas se triplica.

Esta es la forma no recursiva del algoritmo euclidiano.

Algoritmo 2:

Sea a 412 y b 260.

Vemos que se realiza lo siguiente:

a b r

412 260 108 Vuelta 1

260 108 44 Vuelta 2

108 44 20 Vuelta 3

44 20 4 Vuelta 4

20 4 0 Vuelta 5

Dado que en este algoritmo se hace uso del menor resto no podemos ver la diferencia, pero este algoritmo da una vuelta menos que el primero si el primero no usara el menor resto.

Ahora voy a explicar el cómo funciona:

Sea a, b, c y d constantes pertenecientes al conjunto de los reales

0 a b c

d d

Imagen 1

Podemos ver que el mod (b,d) es c pero podemos observar también que si b fuera negativo se consideraría a como su módulo, si tenemos que ambos son su módulo y se necesita llegar más rápido a un moduló que se sea 0 es mejor tomar a como el modulo aun cuando b es positivo.

Ejemplo:

mod( 8 , 3 ) = 2

mod(-8 , 3 ) = 1

En ambos casos el mcd de -8 y 8 con 3 sale 1, pero usando el primer módulo se necesita verificar una vez más el mod.

Explicación de: a c

If (r>b/2)

r=b-r; b/2 Imagen 2

Donde a+c = b, y vemos que

Si r es c entonces r = b-c, lo que es a.

Algoritmo 3:

Sea a 412 y b 260.

Vemos que se realiza lo siguiente:

a b r

412 260 108 Vuelta 1

260 108 44 Vuelta 2

108 44 20 Vuelta 3

44 20 4 Vuelta 4

20 4 0 Vuelta 5

La diferencia con respecto a los anteriores algoritmos radica en el hecho de que este es un algoritmo recursivo, y al ser recursivo es necesario hallar el n-1, gastando espacio de memoria por eso este es un algoritmo peor que los anteriores, pues hace lo mismo gastando más memoria.(en este también hice uso del menor resto)

Vemos que el algoritmo 3.5 hace lo mismo que el algoritmo 1, pero sin preguntar si r es 0 cuando en el while ya preguntamos por eso por lo que lo sobreentendemos como mejor.

Algoritmo 4:

Sea a 412 y b 260.

Vemos que se realiza lo siguiente:

a b

412 260 Vuelta 1

206 130 Vuelta 2

103 65 Vuelta 3

65 19 Vuelta 4

23 19 Vuelta 5

2 19 Vuelta 5

19 2 Vuelta 5

19 1 Vuelta 6

9 1 Vuelta 7

4 1 Vuelta 8

2 1 Vuelta 9

1 1 Vuelta 10

0 1 Vuelta 11

1 0 Vuelta 12

Este algoritmo hace uso de las propiedades:

-mod(a,b) = (a/2,b)

-mod(a,b) = (a,b/2)

-mod(a,b) = (|a-b|/2,b)

Porque son estas propiedades fundamentales para encontrar el mcd.

* El dividir un numero en binario entre 2 es un corrimiento(tal como en decimal el dividir entre 10 es un corrimiento)
* El corrimiento es una operación muy sencilla.

El uso de estas propiedades hace que todas las operaciones que realicemos sean corrimientos y preguntas, por esto es uno de los más rápidos, pero es recursivo.

Algoritmo 5:

Sea a 412 y b 260.

Vemos que se realiza lo siguiente:

a b

206 130 Vuelta 1

103 65 Vuelta 2

65 19 Vuelta 3

23 19 Vuelta 4

2 19 Vuelta 5

19 2 Vuelta 6

19 1 Vuelta 7

9 1 Vuelta 8

4 1 Vuelta 9

2 1 Vuelta 10

1 1 Vuelta 11

0 1 Vuelta 12

Este consta de 2 partes, en la primera crea un g y le asigna 1, luego se le multiplica \*2 cuantas veces a y b sean divisibles por 2.En la segunda se usan las mismas propiedades de algoritmo 5, pero en vez de hacer solo uso de uno de estos cada vuelta, este algoritmo usa dos. Ya sea a/2 o b/2, y |a-b|/2 creando un par para la siguiente vuelta (con esto aseguramos el a/2 o el b/2)

Porque este es más mejor que algoritmo 4:

* No es recursivo (usa menos memoria)

También podemos decir que hace más procesos por vuelta, pero para evaluar ello tenemos que ver la complejidad computacional de ambos procesos lo cual está fuera del curso.

Algoritmo 5:

Sea a 412 y b 260.

Vemos que se realiza lo siguiente:

a b

412 260 Vuelta 1

152 260 Vuelta 2

152 108 Vuelta 3

44 108 Vuelta 4

44 64 Vuelta 5

44 20 Vuelta 5

24 20 Vuelta 5

4 20 Vuelta 5

4 16 Vuelta 5

4 12 Vuelta 5

4 8 Vuelta 5

El siguiente algoritmo intenta llegar a una igualdad por medio de resta, lo cual es más lento que la división pero, en los casos que estos números no disten mucho de ellos o sean muy pequeños puede ser más efectivo que los demás, esto lo veremos más adelante.

Estadísticas:

Para sacar los datos se debe tener en cuenta lo siguiente:

* Para poder ver un tiempo mayor a 0,00x se llamó a la función 1,000,000 veces(esto porque haciendo uso de la librería ctime no se puede visualizar valores muy pequeños)
* El promedio se realizó de 20 muestras del millón de llamadas a la función.(usando una sumatoria y luego una división entre 20)
* Los valores se recolectaron usando fstream y ctime.
* No se implementó el corrimiento de los algoritmos orientados a el sistema binario(el 4 y 5) ni el uso de la librería NTL

Para las muestras 412, 260 tenemos los siguientes resultados:

Imagen 3

Para las muestras 2934856,239 tenemos los siguientes resultados:

Imagen 4

En la siguiente imagen se a optado por no incluir el valor del algoritmo 6, porque no se puede visualizar las proporciones de las demás algoritmos. El valor del algoritmo 6 es 29.6325, por esto es muy grande para ser comparado con los demás.

Para las muestras 8, 2 tenemos los siguientes resultados:

Imagen 5

Resultados

Los grupos que podemos comparar:

Primer grupo: (Algoritmos 1, 2 ,3)

* Algoritmo 1 es mejor en los 2 primeros casos

Esto es porque a diferencia del algoritmo 2 este tiene un if menos, y en los 3 casos (y por la falta de implementación de NTL y bitset)

* Algoritmo 2 no es mejor que el 1 en ningún caso
* Algoritmo 3 es el peor de los tres pero el mejor en el último caso

En el último caso ha sido el mejor porque es un número pequeño, por tanto solo sirve cuando el número es muy pequeño.

Segundo grupo: (Algoritmo 4 ,5)

* Algoritmo 4 es mejor en todos los casos

Es inesperado esto dado que este es un algoritmo recursivo, por lo que usa una mayor cantidad de memoria, puede que esto se deba a que el algoritmo 5 realiza más pasos por vuelta.

* Algoritmo 5 Es peor en todos los casos

Algoritmo 6 es el mejor en los casos 1 y 3, esto porque este algoritmo realiza una operación de resta, esto hace que en los casos en el que los números sean pequeños este sea un buen algoritmo.

Pero en nuestro caso, vamos a usar números extremadamente grandes, por lo tanto este algoritmo no podrá ser de ayuda para la realización del curso.

Las 20 muestras de cada algoritmo se puede ver en las carpetas “Muestra1”, “Muestra2”,”Muestra3”.