UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

TESIS DE GRADO

COMPARACIÓN Y REFINAMIENTO DE ALGORITMOS DE FACTOORIZACIÓN DE NÚMEROS ENTEROS

Tesis de Grado para obtener el Título de Licenciatura en Informática

Mención Ciencias de la Computación

POR: ROLANDO TROCHE VENEGAS

TUTOR METODOLÓGICO: MsC. GROVER RODRIGUEZ

ASESOR: M.SC. JORGE TERAN POMIER

LA PAZ – BOLIVIA
Junio, 2022

CAPITULO 1. ANTECEDENTES GENERALES O MARCO REFERENCIAL

1.1. Introducción

Factorizar números enteros es importante, hoy más que nunca la factorización de números enteros juega un papel crucial en la vida de todas las personas. Los métodos de encriptación actuales se basan, en gran medida, en la complejidad y el tiempo que toma factorizar números grandes.

También se debe notar la definición de número grande, si se le pregunta a un niño que es un número grande este puede decirnos que es el 100 o hasta el 1000 y si se le preguntara a una persona adulta esta podría decirnos 1 000 000 o 100 000 000, pero esto no es nada si se piensa en los problemas que hoy en día lidian los matemáticos.

Factorizar un número se refiere a encontrar todos los factores primos por los que está compuesto dicho número. El teorema fundamental de la aritmética nos dice que para todo número entero positivo mayor a 1 es un numero primo o bien un único producto de números primos (Euclides, 300 AC).

$$n=\prod_{i=1}^{k} p_i^{\alpha_i}$$

Este es el objetivo que buscamos, encontrar esos factores para números, ahora sí, grandes.

Los algoritmos de encriptación actuales, como RSA, usan números de, por ejemplo, 250 dígitos (829 bits) que fue el último en ser factorizado febrero de 2020. Estos son los números grandes que nos interesan. Los números RSA por ejemplo son números con exactamente dos factores primos, estos números primos, de al menos, la mitad de cantidad de dígitos que el resultado.

Hoy en día tenemos diferentes métodos de factorización como los algoritmos simples de factorización, fracciones continuas, curva elíptica, algoritmos de cribado y hasta nuevos algoritmos basados en programación cuántica.

En este trabajo se estudiará los diferentes métodos de factorización de números y su implementación con diferentes enfoques y refinamientos de implementación que el autor propondrá para ciertos métodos. Con esto se podrá revisar los alcances de los algoritmos de encriptación actuales y a donde se puede llegar en base a estos y realizar una comparación de los mismo, mostrando que algoritmo es el adecuado dada ciertas condiciones.

1.2. Problema

1.2.1. Antecedentes

1.2.2. Planteamiento del problema

Con el avance de las computadoras y el mayor poder de cómputo, toda la seguridad basada en factorización y factores primos corre riesgo de quedar deprecada algún día.

Por esto es necesario tener un compendio de gran parte de los métodos y algoritmos de factorización existentes y sus limitaciones, con el poder

de cómputo actual. Por otro lado, hoy en día con la ayuda de los nuevos y mejores procesadores muchos algoritmos pueden ser paralelizados lo que reduciría su tiempo de ejecución.

1.2.1. Formulación del problema

1.3. Objetivos

1.3.1. Objetivo General

Estudiar y realizar una evaluación de números enteros con diferentes caracterizticas y aplicando sobre ellos algoritmos de factorización de números enteros, comparando el tiempo de ejecución, espacio en memoria y factores primos encontrados.

1.3.2. Objetivos Especificos

- Revisar el estado del arte referido a algoritmos de factorización de números enteros.
- Programar algoritmos de factorización de números enteros.
- Refinamiento de métodos de factorización de números enteros
- Evaluar el desempeño de los programas con base en tiempo de ejecución, espacio en memoria y factores primos encontrados.

1.4. Hipótesis

Los métodos de cribado son los mejores en general para cualquier tipo de número, mientras otros como el método de Fermat responde mejor cuando la distancia entre los factores primos es pequeña.

1.5. Justificaciones

1.5.1. Justificación Tecnológica

1.5.2. Justificación Social

La sociedad en su conjunto se beneficia indirectamente ya que la presente investigación está enfocada al área teórica pero enteramente ligada a futuros campos de investigación y aplicación para mejorar el estilo de vida, hambre de conocimientos y experimentación por parte de la población en general.

1.5.3. Justificación Económica

Con el constante avance de la tecnología y la ciencia en diferentes campos hace que cada día se necesite de mejores equipos, maquinaria, software, hardware entre otros, lo cual implica costos gigantescos, por lo cual hacer una redefinición en las herramientas teóricas resulta ser un gran ahorro y reducción de gastos para las diferentes investigaciones realizadas por universidades, instituciones del estado, comunidades científicas.

Además de una reducción de tiempos, el cual es representado a su vez en una reducción de costos, mejorando así la accesibilidad a nuevos campos de investigación.

1.5.4. Justificación Científica

La presente investigación brinda al campo científico tecnológico un complejo análisis de diferentes algoritmos de factorización, lo cual conlleva a tener nuevos y/o actualizaciones de los mismos generando avances en diferentes campos no solo del área sino también de otros campos afines, generando propuestas o aplicaciones de la presente investigación.

Por lo tanto, al emprender la investigación de los algoritmos de factorización de números enteros grandes y su evaluación, profundiza el conocimiento, aportando así a futuras investigaciones, ya que se está trabajando en un área en desarrollo. Así también como bases prácticas y teóricas para la aplicación de dichos algoritmos en áreas como el análisis complejo de números primos, representación del conocimiento, teoría de números, criptografía, combinatoria, entre otros

1.6. Alcances y Limites

1.6.1. Alcances

- Implementación de los métodos
- Recolección de datos
- Demostración de los métodos
- Paralelización de algoritmos
- Refinamiento de algoritmos
- Comparativa de datos

1.6.1. Limites

- Encontrar factores primos para números RSA grandes
- Romper seguridad actual basada en primos
- Encontrar nuevos números primos
- Algoritmos no estándares

1.7. Metodología

Para realizar la presente investigación se basará en el tipo descriptivo, empleando el método lógico partiendo de casos particulares de

factorización hasta llegar a generalización de las mismas, dando a su vez solución a varias técnicas de factorización.

CAPITULO 2. MARCO TEORICO

En el desarrollo de este capítulo se plantea la teoría relacionada con divisibilidad, números primos, factorización, congruencia, diferentes definiciones en teoría de números que seran fundamentales para el desarrollo de los posteriores algoritmos.

Definiciones e información son recopiladas principalmente de ()

2.1. Números enteros

(Niven, Zuckerman & Montgomery, 1991) La teoría de números se ocupa de las propiedades de los números naturales $1,2,3,4,\cdots$, también llamados enteros positivos. Estos números, juntos con los enteros negativos y cero, forman el conjunto de números enteros.

Para el desarrollo de las siguientes definiciones y algoritmos usaremos la notación de conjuntos para referirnos a los números naturales $\mathbb{N}=\{1,2,3,4,\cdots\}$ y números enteros $\mathbb{Z}=\{\cdots,-2,-1,0,1,2,\cdots\}$ y usualmente si se habla de “número” nos referiremos a un número entero.

2.2. Divisibilidad

(G. H. Hardy & E. M. Wright, 1975, 1975) Un entero a se dice que es divisible por otro entero b, no 0, si es que existe un tercer entero c tal que $a=bc$

Si a y b son positivos, c es necesariamente positivo. Se expresa el echo de que a es divisible por b , o b es un divisor de a , con $b|a$

Por lo tanto $1|a$, $a|a$; y que $b|0$ para cualquier b excepto 0.

También se puede decir que $b \nmid a$ para expresar lo contrario a $b|a$

Otras propiedades que podemos notar son:

$b|a \wedge c|b \rightarrow c|a$

$b|a \rightarrow bc|ac$

Si $c \neq 0$ y $c|a \wedge c|b \rightarrow c|ma+nb$ para todos los enteros m y n

(S. S. Wagstaff, Jr., 2013) Definimos " $n \bmod m$ " como el resto cuando el entero n es divido por el entero positivo m . Siempre se tiene que $0 \leq n \bmod m < m$

Si al menos uno de los enteros n , m no es cero, se define el máximo común divisor (greatest common divisor) de m y n , como gcd(m,n), al mayor entero que divida a ambos m y n. Esta claro que gcd(m,n) >= 1 y que gcd(m,n) = gcd(n,m). Se dice que los enteros m, n son primos relativos o coprimos si gcd(m,n) = 1.

El siguiente enuunciiado es util para los algoritmos que usaremos más adelante, si m es un entero positivo y n, q, r son enteros tal que n = mq + r, entonces gcd(n,m) = gcd(m,r) y su demostración es la siguiente:

Si a = gcd(n,m) y b=gcd(m,r). Dado que a divide a ambos n y m, tambien debe dividir a r =n-mq. Estoo muestra que a es un comun divisor de m y r, entonces este debe ser <= b, su maximo comun divisor. Igualmente, dado que b divide a ambos m y r, este debe dividor a n, entonces b <= a = gcd(n,m). Por lo tanto a = b.

2.2.1. Algoritmo de Euclides

El algoritmo de Euclides es el algoritmo eficiente más antiguo que se tiene, se lo usa desde hace 2500 años, descrito por primera vez por Euclides en su obra Elementos.

Para computar gcd(m,n), con m>=n>0, el algoritmo repetidamente remplaza el par (m,n) por el par (n, m mod n) hasta n=0, en ese punto m es el máximo común divisor que se buscaba.

Algoritmo 1: Algoritmo de Euclides

Entrada: Enteros $m >= n > 0$;
while $n > 0$ **do**
    $r \leftarrow m \bmod n$;
    $m \leftarrow n$;
    $n \leftarrow r$;
end
Salida: $gcd(m, n) =$ el valor final de m;

2.2.2. Algoritmo extendido de Euclides

Si m y n son enteros y al menos unoo de ellos no es 0, entonces existen enteros x y y tales que mx + nx = gcd(m, n)

Esto es importante para caulcular el inverso modular m y para calcular estos valores x, y se puede usar el algoritmo extendido de Eucliides.

Para el siguiente algoritmo se usaraa tripletas de enteros, como u = (u0, u1, u2), las cuales son sumadas y multiplicadas por enteres usando las reglas de suuma vectooriiaal y multiplicacion escalar.

Algoritmo 2: Algoritmo extendido de Euclides

Entrada: Enteros $m >= n > 0$;
$\vec{u} \leftarrow (m, 1, 0)$;
$\vec{v} \leftarrow (n, 0, 1)$;
while $v_0 > 0$ **do**
  $q \leftarrow \lfloor u_0/v_0 \rfloor$;
  $\vec{w} \leftarrow \vec{u} - q\vec{v}$;
  $\vec{u} \leftarrow \vec{v}$;
  $\vec{v} \leftarrow \vec{w}$;
end
Salida: $(gcd(m, n), x, y) =$ el valor final de $\vec{u}$.

Los números x y y no son únicos, pero el algoritmo retorna los x y y con valor absoluto más pequeños. El algoritmo funciona porque cada tripleta (a,b,c) satisface a = bm + cn duurante el algoritmo.

2.3. Números primos

Un entero p > 1 es llamado número primo, o simplemente primo, si no exite un divisor d de p que satisfaga 1 < d < p. Si un entero a > 1 no es un primo, es llamado un número compuesto.

2.4. Teorema fundamental de la aritmética

Todos los enteros mayores a 1 se pueden escribir como un producto de primos, este producto es único aparte del orden de los factores primos, y a este producto se llamara la factorizacion de dicho numero.

N = p1 x p2 x p3 x p4

Si se tiene e copias de un primo p en el producto, este se puede reescribir como p^e, de esta forma se obtiene la forma canonica de la factorizacion de un entero n como

Forma canonica factorización

donde p1, p2, …, pk son los distintos primos que dividen a n y ei >=1 es el umero de veces que pi divide a n. Si n es primo, solo existe un "factor", el mismo primo. Tambien esta permitido n=1 con el "producto vacio" para esta forma canonica

2.5. Congruencia

Si m es un entero positivo y a y b son enteros, se puede decir a que a es congruente coon b moduulo m y escribir a = b(mod m) si m divide a a-b. Si m !| *a-b) se esceivre a!=b (mood m). El entero m es llamado el modulo. Cuando a= b (mod m), cada uno a yb son llamados resuuduos del otro (modulooo m). Congurencia modulo m es una relacion de equivalencia, lo que significa que si a,b y c son enteros, entonces

- a = a modm
- Si a = b modm, entonces b = a mod m
- Si a = b mood m y b = c mod m entonces a = c mod m

La congruencia a = b mod m es equivalente a decir que existe un entero k tal que a = b + km

2.6. Notación Big O

2.7. Divisiones sucesivas

Si se quiere encontrar los factores primos de n lo que se hace es tener una lista de todos los números primos menores a n, luego se prueba dividir n entre cada primo p_i, si $p_i \vee n$ entonces se vuelve a hacer el proceso desde ese primo, pero ahora con n/p_i. Si se llega a $\sqrt{n}$ y no se

ha encontrado ningún primo que divida a n, entonces se declara a n como primo.

2.8. Método de diferencia de cuadrados de Fermat

Para factorizar un número impar n, Fermat trato de expresar n como una diferencia de dos cuadrados, $x^2 - y^2$ con el par x, y diferente de $\frac{n+1}{2}, \frac{n-1}{2}$. Este par entrega $x + y = n$ y $x - y = 1$. Cualquier otra representación de n como $x^2 - y^2$ entrega una factorización no trivial $n = (x - y)(x + y)$

Algoritmo

Entrada: Un entero compuesto impar *n*

x = $\sqrt{n}$

t = 2x + 1

r = x^2 – n

while(r no sea una raíz cuadrada) {

r = r + t

t = t + 2

}

x = (t-1) / 2

y = $\sqrt{r}$

Salida: Los factores $x - y$ y $x + y$ de n

2.9. Algoritmo de factorización en una línea de Hart

Hart en 2012 invento una variación del Método de Factorización de Fermat, que es mucha más corto, simple de programar. El da un argumento heurístico de que factoriza n en $O(n^{\frac{1}{3+\varepsilon}})$ pasos.

El algoritmo de Hart comienza verificando si n es una raíz. Si n no es una raíz, entonces hace divisiones sucesivas, pero se detiene cuando p alcanza $n^{\frac{1}{3}}$. En caso que n no ha sido factorizado todavía, realiza los siguientes pasos.

Para $i=1,2,3,\ldots$ prueba cualquier $¿\sqrt{n_i}\vee ¿^2 \, mod \, n ¿$ si es raiz. Si este número es igual a t^2 entonces, es un factor de n

Algoritmo después de verificar si es raíz y las divisiones sucesivas

Entrada: Un entero positivo n **y un límite** L

for(i = 1 hasta L){

s = $¿\sqrt{n}\vee ¿$

m = s^2 mod n

if(m es raíz) {break}

}

t = $\sqrt{m}$

Salida: mcd(s-t, n) es un factor de N

Este algoritmo es especialmente rápido para enteros de la forma $(c^a+d)(c^b+e)$ donde $c, |d|, |e|$ y $¿a-b\vee ¿$ son enteros positivos pequeños

2.10. Variación de Fermat de Lehmers

En 1985, Lawrence propuso una manera de factorizar n cuando se cree que $n=pq$ con $p \leq q$, donde la proporción p/q es aproximadamente a/b y

a y b son coprimos pequeños. Cuando $a=b=1$, este algoriitmoo es lo mismo que el de Fermat. Asumiendo que $mcd(ab,n)=1$.

Suponemos primero que ambos a y b sooon impares. Escriba $x=¿\sqrt{abn}\vee ¿$. Se prueban los enteroos $(x-1)^2-abn$, $i=0,1,2,\ldots$ si son una raíz como en Fermat. Se supone que j es el primer valor de i para el cual este número es una raíz, entonces $(x-j)^2-abn=y^2$

Entonces:

$$abn=(x+j)^2-y^2=(x+j+y)(x+j-y)$$

Se remueve los factores de ab de los dos factores del trinomio para obtener los factores de n. Esto es, $mcd(x+j+y,n)$ y $mcd(x+j-y,n)$ serán los factores de n.

Cuando una de los dos a o b es par y el otro es impar, los cálculos son un poco más complicados porque se debe lidiar con mitades. Lehman evito este problema multiplicando a,b y los otros números en el algoritmo por 2

2.11. Método de Pollard Rho

Es un algoritmo de factorización de enteros. Este fue inventado por John Pollard el año 1975. Este no utiliza mucho espacio de memoria, y el tiempo esperado de ejecución es proporcional a la raíz del factor primo más pequeño del número compuesto a ser factorizado. Está basada en la combinación de 2 ideas, que también son útiles para muchos otros métodos de factorización. La primera idea es la bien conocida Paradoja del cumpleaños: un grupo de al menos 23 personas seleccionadas

aleatoriamente contiene 2 personas con el mismo cumpleaños en más del 50% de los casos. Más generalmente: si los números son elegidos de manera aleatoria en un conjunto de p números, la probabilidad de elegir el mismo número dos veces excede el 50% después de $1.177\sqrt{p}$ números elegidos. El primer duplicado se espera que aparezca después de que $c*\sqrt{p}$ hayan sido seleccionados, para algún pequeño constante c

La segunda idea es la siguiente: si p es algún divisor desconocido de n y las variables x, y son 2 enteros que se piensa son idénticas modulo p, en otras palabras $x \equiv y \, mod \, p$, entonces este puede ser verificado calculando $mcd(|x-y|, n)$; más importante, este cálculo puede revelar una factorización de n, a menos que x, y también sean idénticos modulo n.

Estas ideas pueden ser combinadas en un algoritmo de factorización de la siguiente manera.

Generar una secuencia en $\{0, 1, \ldots, n-1\}$ seleccionando x_0 y definiendo a x_{i+1} como el resto no negativo más pequeño de $x_i^2 + 1 \, mod \, n$, ya quee p divide a n los restos no negativos más pequeños $x_i \, mod \, p$ y $x_j \, mod \, p$ son iguales si y solo si x_i y x_j son identicos modulo p, ya que x_i mod p se comporta más o menos como un entero aleatorio en $\{0, 1, \ldots, p-1\}$ podemos esperar factorizar n calculando $mcd(|x_i - x_j|, n)$ para $i \neq j$ después de que al menos $c*\sqrt{p}$ elementos de la secuencia han sido calculados

2.12. Método de p-1 de Pollard

2.13. Fracciones continuas

2.14. Curva Eliptica

2.15. Factorizacion con curva eliptica

2.16. Métodos de cribado