Departamento de Matemática y Ciencia de la Computación

Licenciatura en Ciencia de la Computación

Laboratorio N°3

Factorización de Números por Fuerza Bruta

Introducción a la Criptografía Moderna
Profesor: Rodrigo Abarzúa

Matías Daza A.
María Fernanda Maldonado P.

Índice general

1. Factorización de números enteros

En este laboratorio estudiaremos el problema de determinar la factorización de un entero n. El método más simple es dividir n por primos hasta que un divisor p es encontrado. Entonces se aplica este método para n/p continuand hasta que obtenemos todos los factores primos de n. Este algoritmo necesita un conjunto de números primos que los podemos encontrar con el algoritmo **Sieve de Eratothenes** los cuales son pasos 1),...,4) en el siguiente algoritmo. Suponga que tenemos una tabla de primos hasta n, Entonces podemos completar todos los factores de números hasta n^2. El siguiente algoritmo conocido conocido como **Trial Division** nos permite conocer la factorización prima de un entero n. Dado un entero n, la salida del algoritmo es su factorización. Se usa una lista llamada factors para almacenar la factorización de n de la forma p,e donde p^e es una de las potencias que aparecen al descomponer n.

1.1. Algoritmo

Algorithm 1 Algoritmo de factorización de números por fuerza bruta

Require: int n, i=0, x, Elementos, j, pi, y=0, m, e, q, indice=0
Ensure: $(primos[i])^e$

```
1: n=strtol(argv[1], NULL, 10);
2: m=n
3: n=sqrt(n)
4: int p[n], primos[n];
5: for (i = 0; i <= (n − 2); i + +) do
6:    p[i]=i+2
7: end for
8: Mostrar Valor de raíz de n = n
9: Mostar Valores de P=
10: for (i = 0; i <= (n − 2); i + +) do
11:    Mostrar p[i]
12: end for
13: Elementos = i;
14: for (i = 0; i < Elementos; i + +) do
15:    if (p[i]! = 0) then
16:       pi=p[i];
17:    end if
18:
19:    for (j = 0; j < Elementos; j + +) do
20:       if (p[j]! = 0) then
21:          if (p[j] %pi == 0) then
22:             p[j]=0;
23:          end if
24:       end if
25:    end for
26:    primos[y]=pi
27:    Mostrar primos[y]
28:    y++;
29: end for
30: Elementos=y;
31: for (y = 0; y < Elementos; y + +) do
32:    if (y! = Elementos − 1) then
33:       Mostrar primos[y]
34:    end if
35: end for
36: q=m;
37: for (i = 0; i < Elementos; i + +) do
38:    q=m;
39:    e=0;
40:    if (q %primos[i] == 0) then
41:       while (q %primos[i] == 0) do
42:          e++;
43:          q =(q / primos[i])
44:       end while
45:       if (indice == 0) then
46:          Mostrar primos[i]^e;
47:          indice++;
48:       end if
49:    end if
50: end for
51: if (indice == 0) then
52:    Mostrar El número es primo, por lo que no tiene divisores.
53: end if
```

1.2. Formulación del experimento

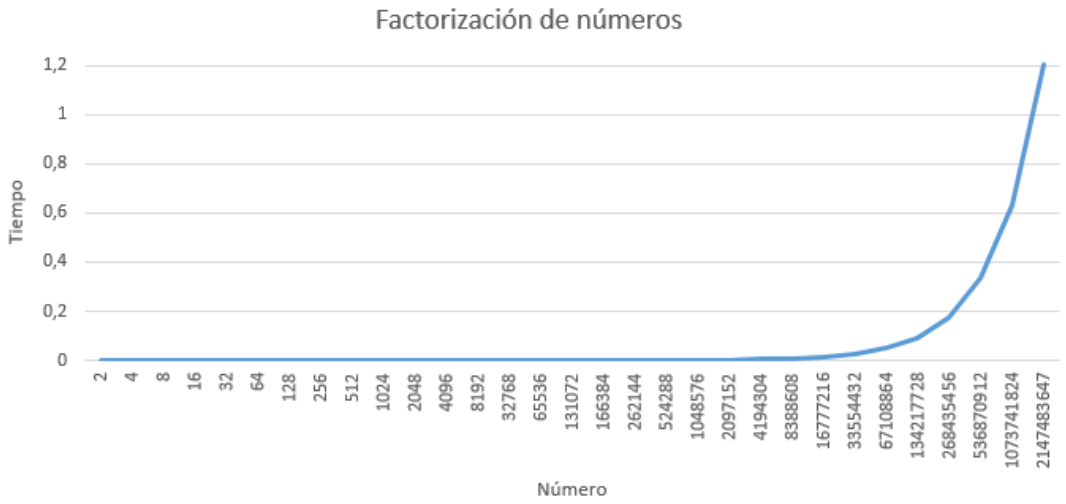
En esta etapa se manejó con arreglos de largo igual a la raíz del número ingresado por el usuario, donde se desarrollaron las siguientes etapas:

1. Llenar el arreglo "p" con los valores correlativos desde el 2 hasta raíz de n.
2. Sacar los valor de "p" que sean primos en el arreglo "$primos$".
3. Finalizar mostrando la cantidad de veces que se repite un número primo como divisor.

En la implementación de este laboratorio se usaró el manejo simple de arreglos ya que por característias de Hardware el uso de gmp no pudo concretarse.

1.3. Análisis del experimento

A continuación se adjunta gráfico de desempeño del algoritmo, de acuerdo a los datos específicados en el archivo "Experimentos.txt".



2. Conclusiones

En este proceso pudimos desarrollar un algoritmo de alto costo computacional, que si bien es cierto para números pequeños no es tan costoso, para números más grandes se vuelve engorroso.

En esta implementación de algoritmo, pudimos hasta el número 2147483647, que es el máximo que soporta nuestro equipos.

3. Estructuras de Datos Utilizadas

Se utilizaron arreglos normales para la representación de los números que descomponen el número entregado por el usuario.

4. Lectura de programa desde la consola

La lectura del programa se indica a continuación:

Para compilar:

- gcc -o -lm Ejecutar Lab3.c

Para ejecutar:

- ./Ejecutar n, siendo n el número que queremos factorizar.

5. Implementación

El algoritmo está implementado en lenguaje C.

6. Plataforma Computacional

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

- **Procesador:** Intel® Core™ i3-350M CPU @ 2.27GHz × 4
- **Memoria RAM:** 5,6 GiB
- **Tipo de SO:** 64 bits
- **Sistema Operativo:** Linux - Ubuntu 14.04 LTS

6.1. Librerías

- stdio.h
- stdlib.h
- math.h