Universidad del Valle de Guatemala
Cifrado de Información - CC 3078 - Sección 10
Julio Herrera 19402
Bryann Alfaro 19372
Diego Arredondo 19422

Funciones Maximo Comun Divisor y Teorema de Fermat
Laboratorio 7

INTRODUCCIÓN

En el presente laboratorio se explorará el funcionamiento del algoritmo de Euclides para poder realizar el cálculo del máximo común divisor entre dos números. De igual manera, se realizará la implementación de la versión extendida de este algoritmo para poder calcular sus coeficientes.

Seguidamente, se implementará el cálculo de inversos modulo n. Por último, se implementará el test de primalidad de Fermat para poder evaluar la posibilidad de que dado un número, este sea primo o no y luego usando esta misma función, realizar la generación de números primos aleatorios.

METODOLOGÍA

Datos

- Parte 1:
  - a1, b1: 1036,240
  - a2, b2: 22896,192
  - a3, b3: 689161,378851
- Parte 2:
  - a1, b1: 1036,240
  - a2, b2: 8753,3354
  - a3, b3: 2021,43
- Parte 3:
  - a1, b1: 47, 2020
  - a2, b2: 31, 1234
  - a3, b3: 65, 17316
- Parte 4:
  - k = 5 (repeticiones de test)
  - Números a *testear*: 1317, 2709, 3257, 3911, 4279, 5497, 6311, 7223, 8431, 9203.

Parte 1:
El algoritmo de Euclides es un método antiguo desarrollado por Euclides para encontrar el Máximo Común Divisor (MCD) de dos números enteros.

Para esta implementación se hace uso de una función recursiva, siguiendo la teoría del método que va reduciendo ambos números dividiendo el mayor entre el menor y si la división es exacta, el divisor (número más pequeño) es el MCD.

Traducir este algoritmo a código de python es sencillo ya que el lenguaje nos ofrece el operador módulo (%) que de entre dos números retorna el residuo de la división del número izquierdo entre el derecho. Si el resultado de esta operación es 0 se devuelve el divisor (número más pequeño), sino, se usa la recursividad volviendo a llamar a la operación ahora con el número más pequeño como primer operando y el resultado como segundo. Como condición de freno a la recursividad y parte del algoritmo de euclides se hace una revisión inicial, si el menor número es igual a cero, entonces el MCD es igual al mayor número.

1. Recibir ambos números, a y b.
2. Obtener el valor absoluto de ambos números.
3. Si b es igual a cero
    a. Entonces a es el MCD
    b. Nota: Aquí se evita la comprobación de si a es igual a cero, ya que se realiza posteriormente por la recursividad.
4. Si no
    a. Aplicar recursividad usando b como primer parámetro y el resultado de la operación módulo (a % b) como segundo parámetro.

Parte 2:
El algoritmo de Euclides extendido es una variante del método original que representa al MCD como una combinación lineal dada por $mcd(a,b) = ax + by$, por lo tanto este método nos permite encontrar tanto los coeficientes como el MCD.

Para comenzar a aplicar este método debemos saber que $mcd(a,b) = mcd(b, a \bmod b)$ como se aplicó en la **Parte 1**, y haciendo que $a_1 = a$, $b_1 = b$, $a_2 = b$ y $b_2 = (a \bmod b) = a - bk$ donde *k* es entero podemos llegar por medio de sustituciones y la suposición de múltiples iteraciones de la operación (recursividad) a que los coeficientes x y y en la última iteración serán 1 y 0 respectivamente.

1. Se reciben ambos números, a y b
2. Se declaran los coeficientes esperados de la última iteración
    a. x_1 = 0
    b. y_1 = 1
3. Como condición de freno a la recursividad: si a es igual a cero
    a. regresar b (MCD), coeficiente x y coeficiente y.
4. Si no
    a. Aplicar recursividad, utilizando b%a como primer parámetro y a como segundo y obteniendo los valores resultantes en mcd, x y y.
    b. Aplicar la ecuación lineal a $x_1 = y - (b / a) * x$.
        i. Nota: Se utiliza el operador // en la división en la forma (a//b) para obtener un resultado entero.

c. $y_1 = x$
d. Retornar mcd, x_1 y y_1

Parte 3:
El inverso multiplicativo es la "división" en la aritmética modular, por lo tanto para un entero proveniente de $a \bmod n$, el inverso multiplicativo es otro entero $c \bmod n$ que es congruente a $1 \bmod n$ o a $(a * a^{-1}) \bmod n = 1$. Una propiedad interesante es que el inverso multiplicativo del entero $a \bmod b$ existe si y sólo si a y b son coprimos ($mcd(a, n) = 1$).

Para realizar este método se hace uso del algoritmo de Euclides extendido visto en la **Parte 2**, donde se sustituirá en la combinación lineal b por n, quedando así $mcd(a, n) = ax * ny$, y donde sabemos que $mcd(a, n) = 1$.

El punto es obtener el MCD resultante y el coeficiente x, de forma que si el MCD es igual a 1, podemos remover el término ny ya que $y = 0$ según el algoritmo extendido de euclides y el módulo inverso es $ax = 1 \bmod n$.

1. Recibir el a y n, de $a \bmod n$
2. Obtener el mcd y el coeficiente x
    a. Nota: El coeficiente $y = 0$ es descartado.
3. Si el mcd no es igual a 1:
    a. Los números a y n no son primos y por lo tanto no existe el inverso.
4. Si no
    a. Regresar el inverso $x \% n$

Parte 4:
El *test* de Fermat es utilizado para saber la probabilidad de si un número es primo a partir de realizar varias iteraciones aplicando la teoría del pequeño teorema de Fermat, que dice que si p es primo y a es coprimo con p, entonces $a^{p-1} = 1 \bmod p$.

Para realizar el *test* de Fermat se hace uso de la exponenciación binaria, un método rápido utilizado para calcular grandes potencias enteras de un número (a^n) usando solo $O(\log n)$ multiplicaciones.

Para el *test* de Fermat generamos un número aleatorio en cada iteración en el intervalo de 2 a n-2 (se omite 1 ya que se sabe que al realizar la exponenciación binaria con este número siempre retornará 1). Luego se llama a la función recursiva de exponenciación binaria (binpow) ingresando como parámetro a el número random y b el número a comprobar - 1. El resultado de esta potencia se le aplica módulo con el mismo número a comprobar, si es diferente a 1 quiere decir que existe otro número por el que se puede dividir esta potencia y por lo tanto no es primo.

Si ninguna iteración con los números random dió falso, entonces el número posiblemente es primo y retornar los números comprobados, si una iteración mostró lo contrario, entonces el número no es primo y retornar el número con el que falló la prueba.

```python
def testFermat(n,k):
    prime = True
    pruebas = []
    for i in range(0,k):
        a = np.random.randint(2,n-2)
        pruebas.append(a)
        if (binpow(a,n-1)%n)!=1:
            prime = False
    if prime:
        return prime,pruebas
    else:
        return prime,a

#https://cp-algorithms.com/algebra/binary-exp.html
def binpow(a,b):
    if b==0:
        return 1
    result = binpow(a,b//2)
    if b % 2:
        return result * result *a
    else:
        return result*result
```

Por último se compara la posibilidad obtenida de cada número con 5 iteraciones, con una tabla real de números primos. Se utilizó la tabla https://primes.utm.edu/lists/small/10000.txt.

Parte 5:
Para crear un generador de números primos aleatorios podemos utilizar el *test* de fermat de la **Parte 4** y generar números aleatorios e irlos comprobando si son primos o no, para así luego hasta tener la cantidad de números requerida y retornar la lista completa de primos encontrados.

1. Se recibe la longitud mínima del número a generar.
2. Realizar las iteraciones necesarias
    a. Generar un número random entre el mínimo y mínimo*100
    b. Usando el *test* de Fermat determinar si es posiblemente primo
        i. Si es posiblemente primo, agregar a la lista.
3. Retornar la lista de primos encontrados.

RESULTADOS

Primera parte:

```
 GDC Euclids Algorithm
4
48
341
```

Segunda parte:

```
 Extended Euclids Algorithm
(4, 19, -82)
(1, -679, 1772)
(43, 0, 1)
```

Tercera parte:

```
 Mod Inverse
43
1035
No existe inverso
```

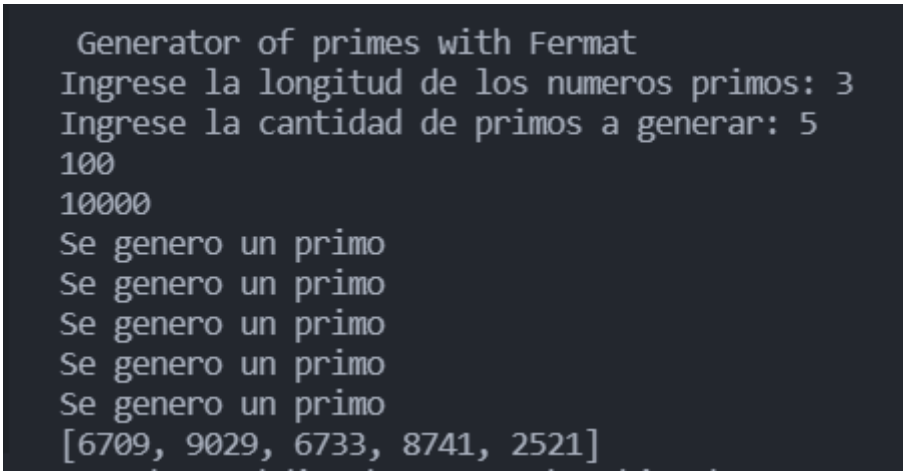
Cuarta parte:

```
 Fermat test
(False, 787)
(False, 950)
(True, [1017, 987, 1222, 599, 2794])
(True, [2063, 267, 1189, 2825, 3421])
(False, 1454)
(False, 2601)
(True, [6121, 2609, 6021, 5101, 871])
(False, 5475)
(True, [3495, 8103, 2094, 5239, 5190])
(True, [4790, 4565, 441, 1434, 1780])
3257 teoricamente, es un numero primo
3911 teoricamente, es un numero primo
6311 teoricamente, es un numero primo
8431 teoricamente, es un numero primo
9203 teoricamente, es un numero primo
```

Comparación:

Número	Posibilidad obtenida	¿Es primo?	¿Acertó?
1317	No	No	Sí
2709	No	No	Sí
3257	Sí	Sí	Sí
3911	Sí	Sí	Sí
4279	No	No	Sí
5497	No	No	Sí
6311	Sí	Sí	Sí
7223	No	No	Sí
8431	Sí	Sí	Sí
9203	Sí	Sí	Sí

Quinta parte:



CONCLUSIONES

1. Dados los resultados, se puede concluir que el test de fermat es bastante efectivo ya que los que “probablemente sean primos”, efectivamente lo eran. Como también es eficaz identificando aquellos números que no lo son.