La funcion φ de Euler o Euler Totient

Matemáticas Discretas 2: 2023-1 | Universidad Nacional De Colombia

Juan Carlos Garavito Higuera

El presente reto se nos muestra para facilitar el análisis y cálculo de generadores en los grupos cíclicos, además de ver la diferencia entre una aproximación que realiza el cálculo a fuerza bruta vs una implementación con el lema de Euler Totient.

DESARROLLO DE LA SOLUCIÓN DEL PROBLEMA.

El acercamiento realizado consiste en un programa modular en el que se toman en cuenta todos los pasos del Euler totient con los grupos cíclicos.

La funcion φ de Euler o Euler Totient se puede definir de la siguiente manera:

$$\varphi(n) = | \{x: 1 <= x <= n \ \wedge mcd(n,x)=1\} |$$

Teorema: Sea G un Grupo cíclico de orden n, generado por a, entonces G tiene φ(n) generadores.

Lema Euler Totient:

i. $\varphi(1) = 1$.

ii. Si p es primo: $\varphi(p^a) = (p^a) - (p^a-1)$.

iii. Si $mcd(m,n) = 1$: $\varphi(mn) = \varphi(m)\ \varphi(n)$.

Las librerias a usar serán math y time.

```python
import math         #Libreria de manejo matemático.
import time         #Libreria de manejo y análisis de tiempo.
```

Comenzamos por definir una función que nos ayude a determinar la primalidad de un número:

```python
def es_primo(numero):
    # Si el número es menor o igual a 1, no es primo
    if numero <= 1:
        return False
    # Si el número es 2 o 3, es primo
    elif numero <= 3:
        return True
    # Si el número es divisible por 2 o 3, no es primo
    elif numero % 2 == 0 or numero % 3 == 0:
        return False
```

```
    # Comienza en 5 y verifica si el número es divisible por cualquier
número de la forma 6k ± 1
    i = 5
    while i * i <= numero:
        if numero % i == 0 or numero % (i + 2) == 0:
            # Si el número es divisible por un número de la forma 6k ±
1, no es primo
            return False
        # Avanza al siguiente número de la forma 6k ± 1
        i += 6
    # Si el número no es divisible por ningún número de la forma 6k ±
1, es primo
    return True
```

Ahora definimos una función que nos ayude a factorizar un número en sus factores primos:

```
def factorizar_primos(numero):
    # Comprobar si el número es primo
    if es_primo(numero):
        return [numero]  # Si el número es primo, devuelve solo el
número de entrada.

    # Inicializar una lista para almacenar los factores primos
    factores_primos = []

    # Iterar hasta factorizar completamente el número
    i = 2  # Comenzar con el número primo más pequeño
    while i <= math.sqrt(numero):  # Iterar hasta la raíz cuadrada del
número para menor carga computacional
        if numero % i == 0:
            factores_primos.append(i)  # Agregar i a la lista de
factores primos
            numero //= i  # Dividir el número por i y actualizar su
valor
        else:
            i += 1

    if numero > 1:  # Si después del bucle el número de entrada es
mayor que 1, significa que quedó algún factor primo
        factores_primos.append(numero)  # Agregar el factor primo
restante a la lista de factores primos

    return factores_primos  # Devolver la lista de factores primos
```

Luego definimos la función euler_totient_lema que nos permite recopilar las anteriores funciones y aplicar el lema de euler totient.

```
# La función euler_totient_lema implementa el lema de Euler para
calcular el número de elementos en un grupo finito.
def euler_totient_lema(OrdenG):
```

```
    # Si el orden del grupo es 1, hay un solo elemento, el elemento
neutro.
    if (OrdenG == 1):
        return OrdenG
    # Si el orden del grupo es primo, se utiliza la fórmula φ(p) = p -
1 para calcular el valor de φ(n) y se devuelve ese valor.
    elif (es_primo(OrdenG)):
        return OrdenG - 1
    # De lo contrario, se utiliza la factorización en primos del orden
del grupo
    # y la fórmula φ(n) = (p^k - p^(k-1)) * ... * (q^m - q^(m-1)) para
calcular el valor de φ(n) y lo devuelve.
    else:
        numero_generadores = 1
        factores_primos = factorizar_primos(OrdenG)

        # Se utiliza un conjunto para eliminar duplicados y obtener
una lista de factores primos únicos
        set_f_p = set(factores_primos)
        set_f_p = list(set_f_p)

        # Se itera a través de la lista de factores primos y se
calcula el valor correspondiente para cada uno de ellos.
        for primo in range (0,len(set_f_p)):
            potencia = factores_primos.count(set_f_p[primo])
            numero_generadores = numero_generadores *
(pow(set_f_p[primo], potencia)-pow(set_f_p[primo], potencia-1))
        return numero_generadores
```

Ahora definimos el procedimiento a fuerza bruta, realizando unas funciones que calculen el MCD de todos los numeros hasta el entero ingresado:

```
def mcd(x, y):
    # Algoritmo de Euclides para encontrar el máximo común divisor de
x e y
    if y == 0:
        return x
    else:
        return mcd(y, x%y)

def generadores_fb(ordenG):
    generadoresFB=0

    # Bucle for para iterar desde 1 hasta ordenG-1 y encontrar los
generadores del grupo
    for i in range(1,ordenG):
        # Verificar si el máximo común divisor entre ordenG e i es 1
        if mcd(ordenG,i)==1:
            # Si el MCD es 1, entonces i es un generador, aumentar el
```

```
contador de generadores
            generadoresFB+=1
    # Retornar el número total de generadores encontrados
    return generadoresFB
```

Finalmente agregamos una función Main para llamar todas las anteriores funciones y medir sus tiempos de ejecución para hacer el análisis comparativo de los dos métodos:

```
def main():
    OrdenG =  int(input("Ingrese el entero a calcular sus generadores:
"))#23597112
    print("\n|G| = " + str(OrdenG))
    if (es_primo(OrdenG) == True):
        print("¿Es primo?: Si \n")
    else:
        print("¿Es primo?: No \n")

    # Euler Totient
    start = time.time()
    print("Euler Totient: "+ str(euler_totient_lema(OrdenG)))
    end = time.time()
    print("Tiempo: "+ str(end - start) + "\n")


    # Fuerza Bruta
    start = time.time()
    print("Fuerza Bruta: "+ str(generadores_fb(OrdenG)))
    end = time.time()
    print("Tiempo: "+ str(end - start))


main()
```

Bibliografía

https://es.wikipedia.org/wiki/Funci%C3%B3n_%CF%86_de_Euler