Российский университет дружбы народов Научный факультет
Математические основы защиты информации и информационной безопасности

Алгоритм Полларда Rho

НФИмд 02-22
Подготовлено студентом:
Елиенис Санчес Родригес.
Преподаватель: Дмитрий Сергеевич

El algoritmo rho de Pollard es un algoritmo especializado de factorización de números enteros. Fue inventado por John Pollard en 1975. Es especialmente efectivo a la hora de factorizar números compuestos que tengan factores pequeños.

El algoritmo rho emplea pues una función módulo n a modo de generador de una secuencia pseudoaleatoria. Hace funcionar una de las secuencias el doble de rápido que la otra, es decir, por cada iteración de una de las copias de la secuencia, la otra hace dos iteraciones. Sea x el estado actual de una secuencia e y el estado actual de la otra. En cada paso se toma el máximo común divisor (MCD) de |x − y| y n. Si este MCD llega a ser n, entonces finaliza el algoritmo con el resultado de fracaso, ya que esto significa que x = y y, por el algoritmo de la liebre y la tortuga, la secuencia ya ha completado su ciclo y seguir más allá sólo conseguiría repetir trabajo ya realizado

Entrada: Un número entero "n" que no sea una potencia prima.
Salida : Factor no trivial de «n»
a ← 2;
b ← 2;
PARA (i = 1, 2, . . .);
a ← a2 + 1 mod n;
b ← b2 + 1 mod n;
b ← b2 + 1 mod n;
d ← mcd(abs(a – b), n);
SI (1 < d < n)
Retornar d;
SI (d >= n)
Retornar Sin exito;
FIN PARA;
//Observación: Una potencia prima es una potencia entera y positiva de un
// número primo. Por ejemplo $5=5^1$, $9=3^2$ son potencias primas,
//mientras que $6=2\times3$, $15=3\times5$ y $36=6^2=2^2\times3^2$ no lo son.

Implementación:

La función rho usa como función auxiliar una llamada mcd, puedes usar cualquiera que ya hallas implementado

```
import sympy
from colorama import Fore
import math

# функция для генерации основные факторы
def pollard(n):
    #  определяющая основа
    a = 2

    # определяющий показатель
    i = 2

    # повторяйте до тех пор, пока не будет получен простой
множитель
    while (True):

        # # вычисление a по мере необходимости
        a = (a ** i) % n

        # нахождение gcd из a-1 и n
        # использование математической функции
        d = math.gcd((a - 1), n)

        # проверьте, получен ли коэффициент
        if (d > 1):
            # вернуть коэффициент
            return d

            break

        # еще увеличьте показатель на единицу
        # для следующего раунда
        i += 1

print("метод Полларда для разложения на множители")
print()
print("номер, рекомендованный преподаватель: 1359331" )
n = int(input(Fore.YELLOW+"Introduce un numero: "))


# временное хранение n
num = n

# список для хранения простых множителей
ans = []

# повторяется до тех пор, пока все простые множители
```

```python
# получены
while (True):

    # вызов функции
    d = pollard(num)

    # добавить полученный коэффициент в список
    ans.append(d)

    # уменьшить n
    r = int(num / d)

    # проверьте наличие прайма с помощью sympy
    if (sympy.isprime(r)):

        # получены оба простых множителя
        ans.append(r)

        break

    # reduced n is not prime, so repeat
    else:

        num = r

# распечатайте результат
print("los factores primos de", n, "sin", *ans)
```