



RESOLUCION DE LA HOJA DE EJERCICIOS 8

Escuela Profesional: Ingeniería Informática y de sistemas

Asignatura: Algoritmos Avanzados

Docente: Ing. Hector Eduardo Ugarte Rojas

Alumno: Benjamin Alexander Hualverde Quispe

Codigo: 161367

Semestre: 2021_I

Cusco- Perú

2021

HOJA DE EJERCICIOS 8 – ALGORITMOS AVANZADOS UNSAAC

Ejercicio 1:

A. Implementa un código que verifique si dos números son co-primos o no.

```

Numeros Co-Primos.py - C:\Users\Usuario\Desktop\Hoja Ejercicios 8\Numeros Co-Primos.py (3.9.6)
File Edit Format Run Options Window Help

#Modulos para hallar si dos numeros son co-primos

#MCD(para numeros enteros positivos)
def mcd(a,b):
    #Hallar el menor entre ambos
    mini=min(a,b)
    i=mini
    encontrado=False
    divisor=1
    #Hallamos el maximo comun divisor entre ambos numeros:
    while(i>0 and not(encontrado)):
        if(a%i==0 and b%i==0):
            divisor=i
            encontrado=True
        i=i-1
    return divisor

#Determinar si dos numeros son Co-Primos
#(donde a,b son numeros enteros positivos)
def sonCoPrimos(a,b):
    if(mcd(a,b)==1):
        return True
    return False

#Ejemplo:
print("DETERMINAR SI DOS NUMEROS SON co-Primos\n\n")
A=int(input("Digite un numero(entero positivo) : "))
B=int(input("Digite otro numero(entero positivo) : "))
print()
Resultado=sonCoPrimos(A,B)
print("Los dos numeros SI son co-Primos"*Resultado+"Los dos numeros NO son

```

El resultado del programa para un ejemplo, es el siguiente:

```

=== RESTART: C:\Users\Usuario\Desktop\Hoja Ej
DETERMINAR SI DOS NUMEROS SON co-Primos

```

```

Digite un numero(entero positivo) : 23
Digite otro numero(entero positivo) : 25

```

```

Los dos numeros SI son co-Primos

```

```

>>>

```

```

=== RESTART: C:\Users\Usuario\Desktop\Hoja Ej
DETERMINAR SI DOS NUMEROS SON co-Primos

```

```

Digite un numero(entero positivo) : 24
Digite otro numero(entero positivo) : 30

```

```

Los dos numeros NO son co-Primos

```

```

>>> |

```

- B. Ahora implemente un programa que dado un arreglo de tamaño n , encuentre el número de co-primos, y liste la parejas.

Agregamos un programa principal

```
Arreglo de Numeros Co-Primos.py - C:\Users\Usuario\Desktop\Hoja Ejercicios 8\Arreglo de Numeros Co-Pri...
File Edit Format Run Options Window Help

#Hallar el menor entre ambos
mini=min(a,b)
i=mini
encontrado=False
divisor=1
#Hallamos el maximo comun divisor entre ambos numeros:
while(i>0 and not(encontrado)):
    if(a%i==0 and b%i==0):
        divisor=i
        encontrado=True
    i=i-1
return divisor

#Determinar si dos numeros son Co-Primos
#(donde a,b son numeros enteros positivos)
def sonCoPrimos(a,b):
    if(mcd(a,b)==1):
        return True
    return False

#Programa Principal
entrada=input("Digite los numeros(enteros positivos):")
#extraemos los numeros
numeros=entrada.split()
#hallamos los numeros Co-Primos
coPrimos=[]
for a in numeros:
    for b in numeros:
        if(a!=b and [int(a),int(b)] not in coPrimos and [int(b),int(a)]
            coPrimos.append([int(a),int(b)])

print("Los numeros Co-Primos, son:")
print(coPrimos)
```

Ejecutamos un ejemplo para el programa:

```
= RESTART: C:\Users\Usuario\Desktop\Hoja Ejercic
os.py
Digite los numeros(enteros positivos):1 2 3
Los numeros Co-Primos, son:
[[1, 2], [1, 3], [2, 3]]
>>>
= RESTART: C:\Users\Usuario\Desktop\Hoja Ejercic
os.py
Digite los numeros(enteros positivos):4 8 3 9
Los numeros Co-Primos, son:
[[4, 3], [4, 9], [8, 3], [8, 9]]
>>> |
```


Ejercicio 2:

Implemente un código para mostrar la grafica para los 1000 totientes, Haciendo uso del siguiente código:

```
from math import gcd

def phi(n):
    cantidad = 0
    for k in range(1, n + 1):
        if gcd(n, k) == 1:
            cantidad += 1
    return cantidad
```

El programa es el siguiente:

 Grafica_Totiente_Euler.py - C:\Users\Usuario\Desktop\Hoja Ejercicios 8\Grafica_Totiente_Euler.py (3.9.6)

File Edit Format Run Options Window Help

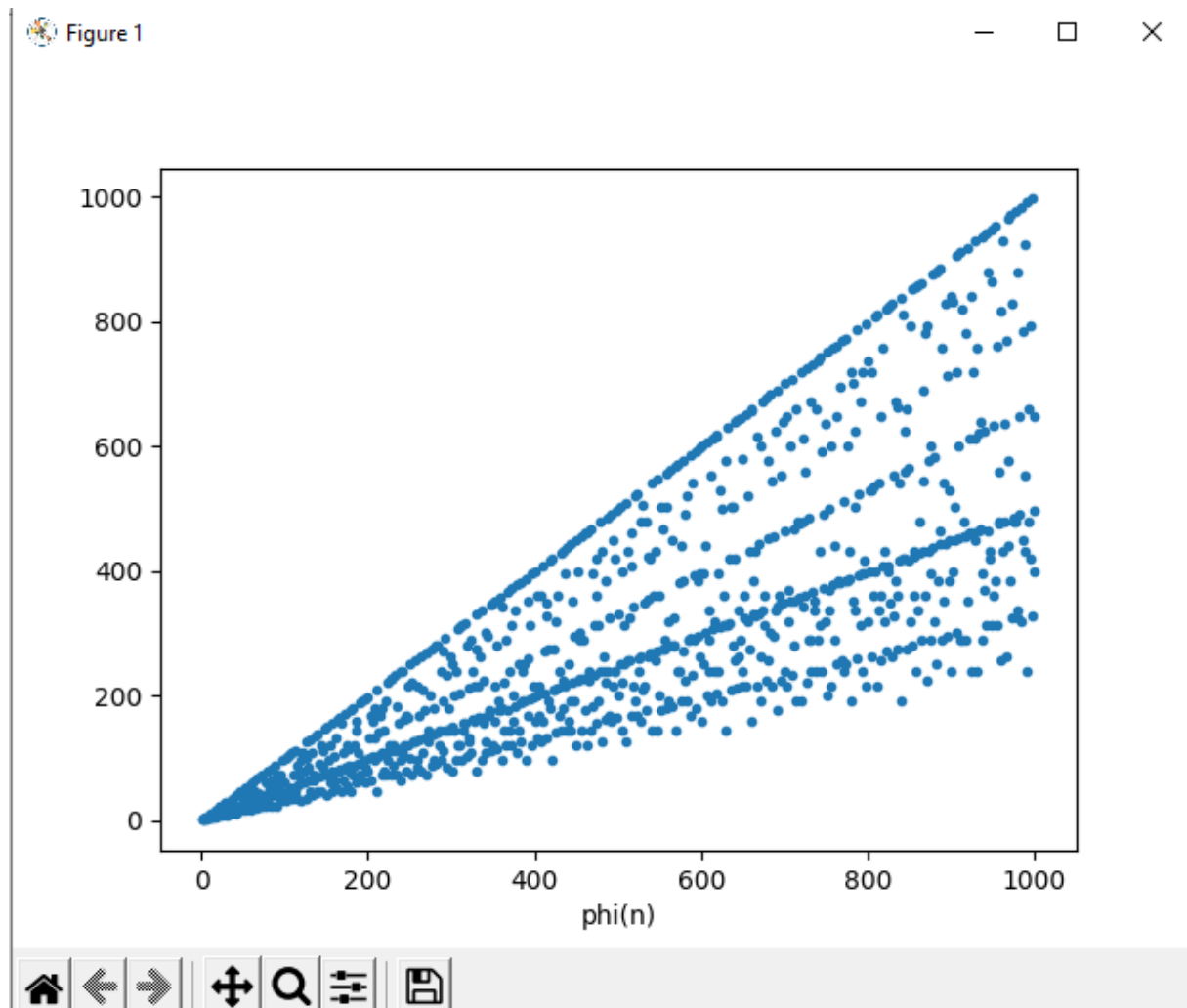
```
import matplotlib.pyplot as plt
from math import gcd
#Mostrar la grafica de para los n primeros totientes de euler:

def phi(n):
    cantidad=0
    for k in range(1,n+1):
        if gcd(n,k)==1:
            cantidad+=1
    return cantidad

#Graficar para los primeros 1000 totientes:
n=[]
phi_n=[]
for i in range(1,1001):
    n.append(i)
    phi_n.append(phi(i))

plt.plot(n,phi_n, '.')
plt.xlabel('n')
plt.ylabel('phi(n)')
plt.show()
```

La grafica resultante es la siguiente:



Ejercicio 3:

Usando el teorema del resto, resuelve a mano el sistema:

$$x \equiv 11 \pmod{5}$$

$$x \equiv 5 \pmod{7}$$

$$x \equiv 7 \pmod{11}$$

Solución:

$$x \equiv 11 \pmod{5}$$

$$x \equiv 5 \pmod{7}$$

$$x \equiv 7 \pmod{11}$$

$$\Rightarrow N = 5 \times 7 \times 11$$

$$= 385$$

b_i	$N_i = \frac{N}{N_i}$	x_i	$b_i N_i x_i$
b_1	77	x_1	?
b_2	55	x_2	?
b_3	35	x_3	?

$$77 x_1 = 1 \pmod{5}$$

$$2 x_1 = 1 \pmod{5}$$

$$x_1 = 3 \pmod{5}$$

$$55 x_2 = 1 \pmod{7}$$

$$6 x_2 = 1 \pmod{7}$$

$$x_2 = 6 \pmod{7}$$

$$35 x_3 = 1 \pmod{11}$$

$$2 x_3 = 1 \pmod{11}$$

$$x_3 = 6 \pmod{11}$$

- Entonces la tabla quedaria:

b_i	N_i	x_i	$b_i N_i x_i$
11	77	3	2541
5	55	6	1650
7	35	6	1470

$$x = 2541 + 1650 + 1470$$

$$x = 5661 \pmod{385}$$

$$x = 271 \pmod{385}$$

: RPTA

$$x = 271$$

Verificación:

$$271 \equiv 11 \pmod{5} \checkmark$$

$$271 \equiv 5 \pmod{7} \checkmark$$

$$271 \equiv 7 \pmod{11} \checkmark$$

Ejercicio 4:

El siguiente código implementa el Teorema Chino del Resto

```
def inverse_mod(a,b):
    x = a
    y = b
    oldolds = 1
    olds = 0
    oldoldt = 0
    oldt = 1
    while y != 0:
        q = x // y
        r = x % y
        x = y
        y = r
        s = oldolds - q * olds
        t = oldoldt - q * oldt
        oldolds = olds
        oldoldt = oldt
        olds = s
        oldt = t
    return oldolds

def chi_rem_thm(mn,an):
    m = 1
    Mn = []
    yn = []
    for k in range(0, len(mn)):
        m = m * mn[k]

    for k in range(0, len(mn)):
        Mk = m / mn[k]
        Mn.append(Mk)
        yk = inverse_mod(Mn[k],mn[k]) % mn[k]
        yn.append(yk)
    x = 0
    for k in range(0, len(yn)):
        x = x + an[k] * Mn[k] * yn[k]
    while x >= m:
        x = x - m
    return x
```

Entiéndalo, coméntalo y verifica el resultado que obtuviste en el ejercicio 3

Después de haber entendido y comentado, agregamos el programa principal para la verificación del resultado del ejercicio 3.

```

Teorema_Chino_Resto_Comentado.py - C:\Users\Usuario\Desktop\Hoja Ejercicios 8\Teorema_Chino_Resto_Comentado.py (...
File Edit Format Run Options Window Help

    yn=[]
    #Hallar el producto tota de los modulos "n"
    for k in range(0, len(mn)):
        m=m*mn[k]
    #Hallar xi para cada ecuacion modular
    for k in range(0, len(mn)):
        Mk=m/mn[k] #En una ecuacion este es el valor de Ni
        Mn.append(Mk)
        yk=inverse_mod(Mn[k], mn[k])%mn[k] #Halla el valor de Xi
        yn.append(yk)
    #Halla la suma total de las multiplicaciones de Ni*Xi*bi
    x=0
    for k in range(0, len(yn)):
        x=x+an[k]*Mn[k]*yn[k]
    #Halla el valor de X
    while x>=m:
        x=x-m
    #Retornar el valor de X
    return x

#=====
#PROGRAMA PRINCIPAL
#Ingresamos los valores del ejercicio anterior
LN=input("Digite n_i de las ecuaciones modulares:")
LN=LN.split()
LN=[int(i) for i in LN]
LB=input("Digite b_i de las ecuaciones modulares:")
LB=LB.split()
LB=[int(i) for i in LB]

#Mostramos las ecuaciones modulares:
print("Las ecuaciones modulares ingresadas, son las siguientes:")
for i in range(len(LN)):
    print(" X = "+str(LB[i])+" mod("+str(LN[i])+")")

#Hallamos el valor de X de las ecuaciones modulares:
print("El valor de X, es:", chi_rem_them(LN, LB))

```

Ingresamos los valores del anterior ejercicio:

```

>>>
= RESTART: C:\Users\Usuario\Desktop\Hoja Ejercicios 8\Teorer
ado.py
Digite n_i de las ecuaciones modulares:5 7 11
Digite b_i de las ecuaciones modulares:11 5 7
Las ecuaciones modulares ingresadas, son las siguientes:
  X = 11 mod(5)
  X = 5 mod(7)
  X = 7 mod(11)
El valor de X, es: 271.0
>>> |

```

Se puede observar que se el valor de X es 271 tal como se llegó en el anterior ejercicio.