Reporte de los algoritmos primo, Fibonacci, fibo, Fibo

Primo

Problemática: saber si el número ingresado es primo o no y saber cuántas operaciones le toma el algoritmo saber si lo es o no, por lo que se tratara de hacer lo más eficiente posible

Descripción del algoritmo:

Definimos primo esto es def primo(n):

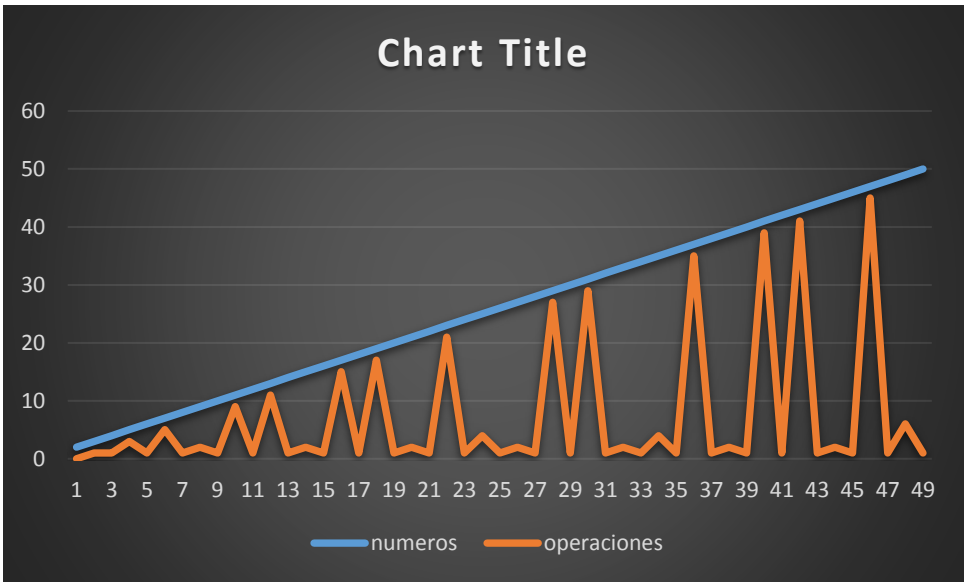
Declaramos nuestro contador como global esto es global cnt

Luego para “i” en el rango de 2 hasta el numero ingresado “n” esto es for i in range(2,n)

#para ser más eficiente se le puede cambiar por (n^ (1/2))

Si el residuo de n entre i es igual a cero entones se imprime (no es primo) y te dice el número que se ingresó y el número de operaciones que realizo es to es if (n% i==0)

De lo contrario te dará el número que se ingresó y el número de operaciones que realizo dando a entender que es primo esto es return n, cnt



Fibonacci

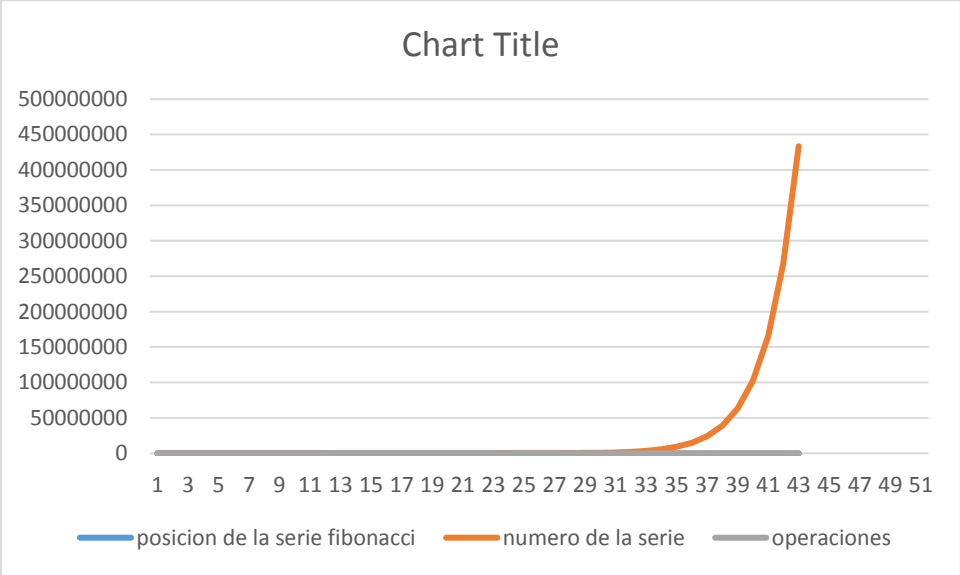
Problemática: obtener el número de la serie Fibonacci ingresando la posición deseada siendo esta posición más grande más tardado será la obtención

Descripción del algoritmo código:

```
cnt=0

def fibonacci(h):
    global cnt
    cnt+=1
    if h==0 or h==1:
        return 1
    return fibonacci(h-2) + fibonacci(h-1)


for i in range(0,51):
        cnt=0
        print(fibonacci(i), cnt)
```

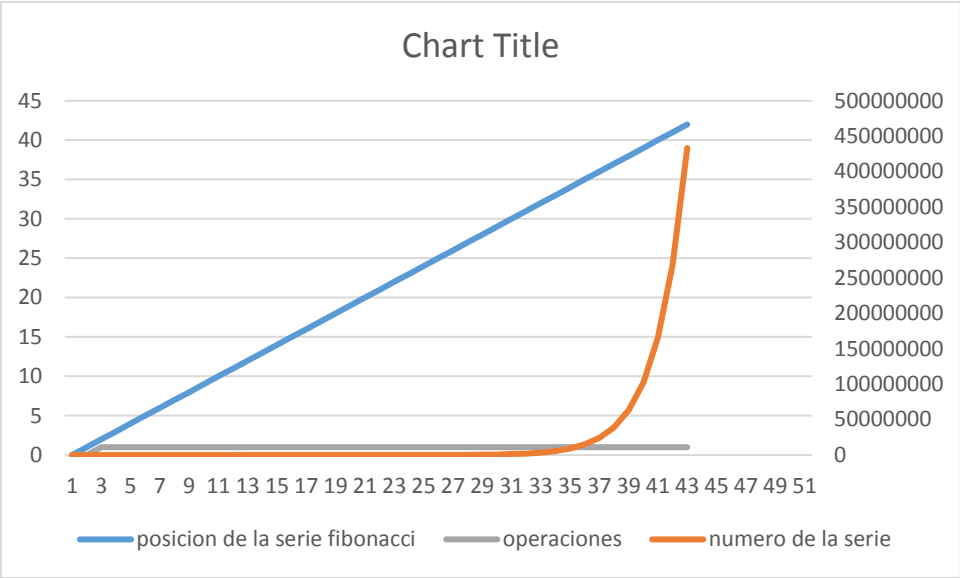


Fibo

Problemática: agilizar o ser más eficientes en el cálculo del número en la posición deseado de la seria Fibonacci

```
cnt=0

def fibo(h):
    w=0
    w1=1
    w2=1
    global cnt
    if h==0 or h==1:
        return 1
    for i in range(2,h+1):
        cnt+=1
        w=w1+w2
        w2=w1
        w1=w
    return w
for h in range (0,43):
    cnt=0
    print(h,fibo(h), cnt)
```



Por lo que vemos el número de operaciones es casi idéntico al número de la posición y obviamente los números de la serie no cambian

Fibonacci con memoria:

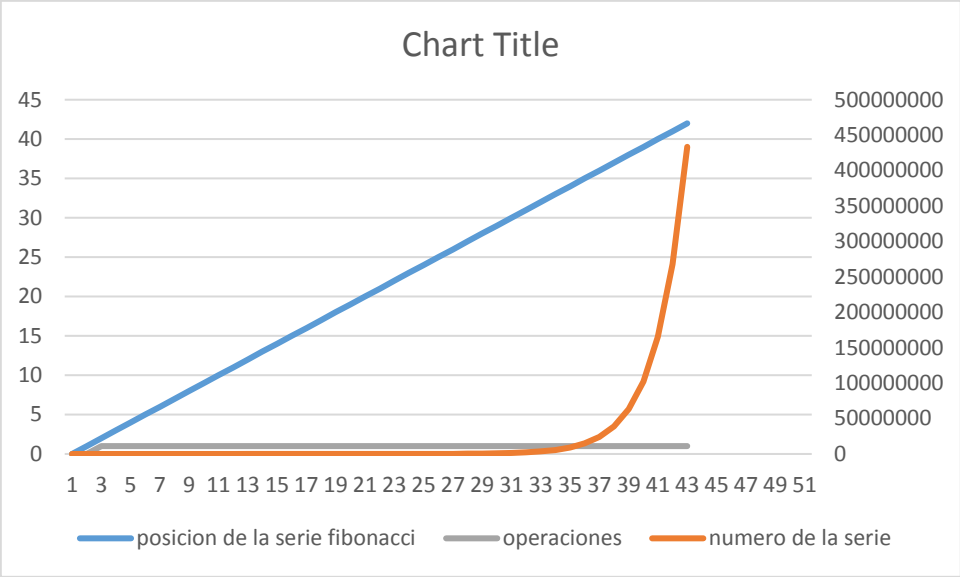
Problemática: obtener los números de la serie Fibonacci que se desee con la ventaja de que el número obtenido se guardara para la próxima vez que se desee obtener el numero haga solo una operación

```
memo={}

cnt=0

def fibonacci(n):
        global memo, cnt
        if n==0 or n==1:
                return 1
        if n in memo:
                return memo[n]
        else:
                cnt+=1
                val=fibonacci(n-2) + fibonacci(n-1)
                memo[n]=val
                return val
```

```
for j in range(0,50):
        cnt=0
        print(j+1, fibonacci(j), cnt)
```



Con esto podemos ver que se puede agilizar la respuesta de saber si un número es primo o no, aunque no hay una forma generalizada de generar un numero primo se puede determinar si es o no uno.

Por lo tanto, con una combinación de fibo y Fibonacci con memoria podemos obtener los números deseados con una velocidad considerablemente mejor.