Villanueva Andrade Daniela Estefanía

1684742

13 Octubre 2017

En ésta etapa programamos algunos algoritmos relacionados a las matemáticas que nos sirven determinar si un número es primo o no, o tratar los tan conocidos números de Fibonacci.

1. **Descripción del problema:** Se quiere determinar si un número cualesquiera es primo o no, para ello fue importante el conocimiento de las condiciones a las que debe obedecer el número para serlo.

   Un número primo sólo es divisible entre 1 y sí mismo.

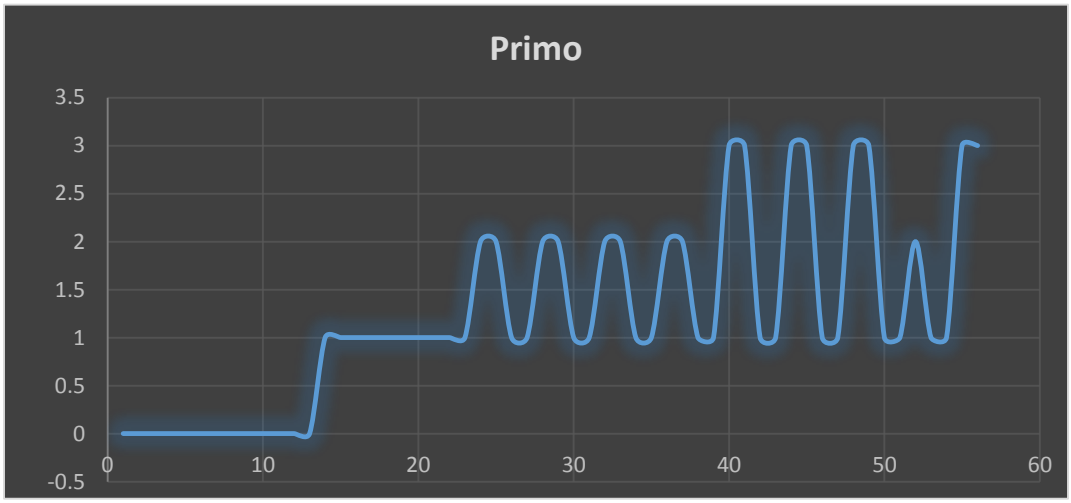
2. **Descripción del algoritmo:**
   Creamos una variable global contador para saber el número de operaciones que se llegan a realizar cada vez que se llama a nuestra función principal PRIMO en la cual determinamos que, si los números menores de la raíz cuadrada de n son divisibles entre ellos quedando como resto cero, entonces no son primos.

```
>>> contador=0
>>> def primo(n):
        global contador
        for i in range(2,round(n**0.5)):
                contador=contador+1
                if((n%i)==0):
                        break
        return contador

>>> for i in range(1,30):
        contador=0
        primo(i)
```

```
        print(contador)
```

Gráfica de desempeño:



1. **Descripción del problema:** partimos de la definición de lo que trata la sucesión de Fibonacci; siendo ésta una sucesión infinita de números enteros donde un número es resultado de la suma de dos de sus anteriores.

   Realizamos diferentes algoritmos para un mismo fin y ver su desempeño y eficacia:

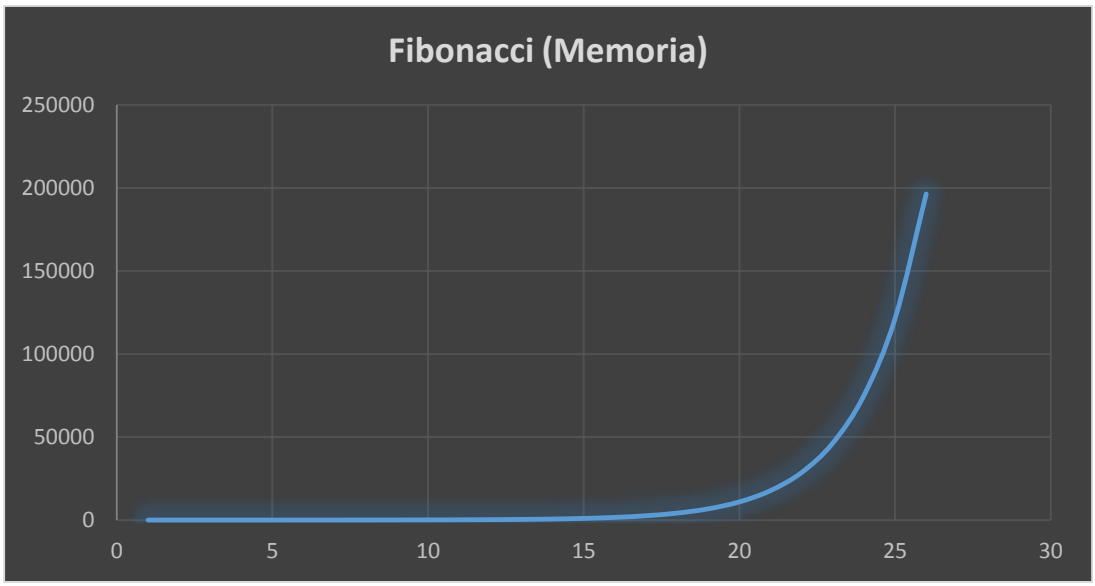
- **Fibonacci (Memoria)**

```
>>> D={}
>>> cnt=0
>>> def fibonacci(n):
        global D, cnt
        cnt+=1
        if n==0 or n==1:
                return(1)
        if n in D:
```

```
            return D[n]
        else:
            val=fibonacci(n-2)+fibonacci(n-1)
            D[n]=val
            return val
```

```
>>> for i in range(1,30):
        cnt=0
        fibonacci(i)
        print(i,cnt)
```

Gráfica de desempeño:



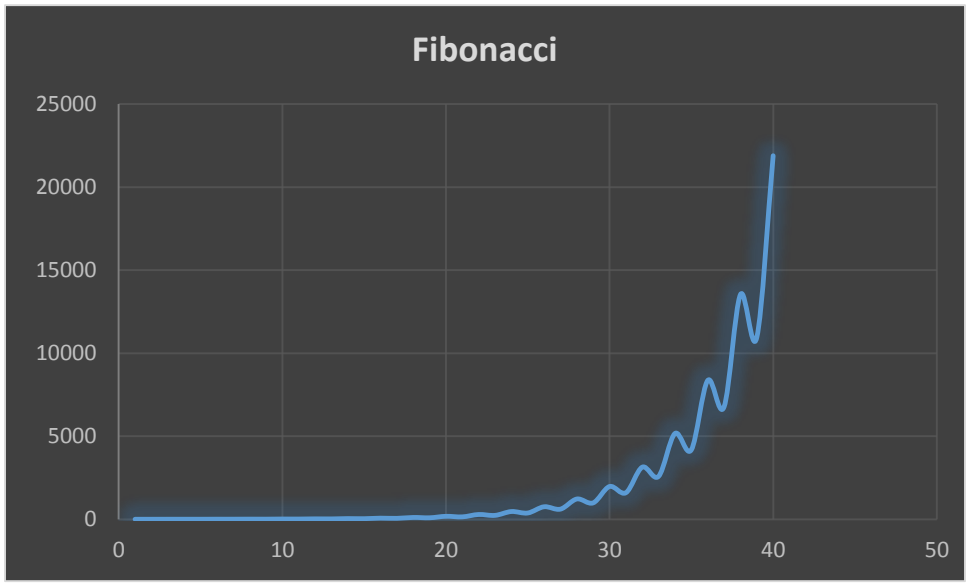
- **Fibonacci**

```
>>> contador=0
>>> def fibonacci(n):
        global contador
        contador+=1
```

```
        if n == 0 or n == 1:
                return(1)
        return fibonacci(n-2)+fibonacci(n-1)

>>> for i in range(1,30):
        contador=0
        fibonacci(i)
        print(contador)
```

Gráfica de desempeño:



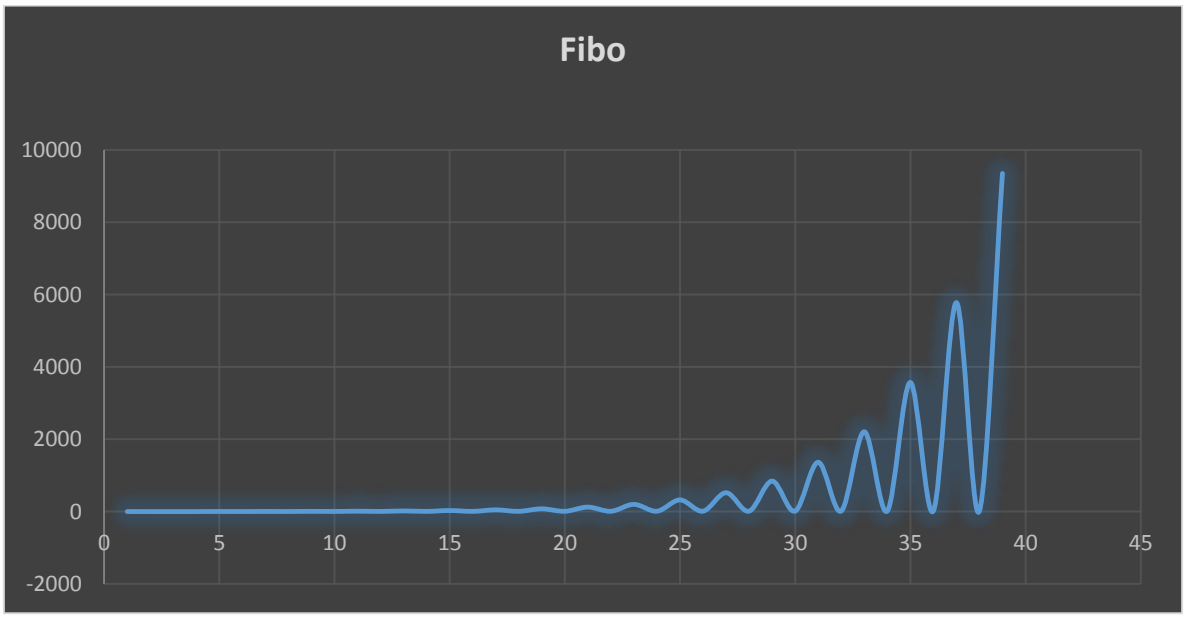
- **Fibo**

```
>>> contador=0
>>> def fibo(n):
        global contador
        if n == 0 or n == 1:
                return(1)
        r,r1,r2 = 0,1,2
        for i in range (2,n):
```

```
            contador+=1
            r=r1+r2
            r2=r1
            r1=r
        return(r)

>>> for i in range(1,30):
        contador=0
        fibo(i)
        print(contador)
```

Gráfica de desempeño:



Conclusión:

En la anterior etapa se pudo notar como interpretar distintos procesos matemáticos en la programación para determinar si un número es primo; emplear los conocimientos para observar la sucesión Fibonacci encontrando aún más aplicaciones con lo que hemos aprendido hasta ahora; crear funciones y empezar a identificar cuál método es el más eficiente.