

Matemáticas computacionales

Reporte de primos y Fibonacci

Alumno: Isaac Emanuel Segovia Olay

M. 1748957

Maestro: Lic. José Anastacio Hernández Saldaña

13-octubre-2017

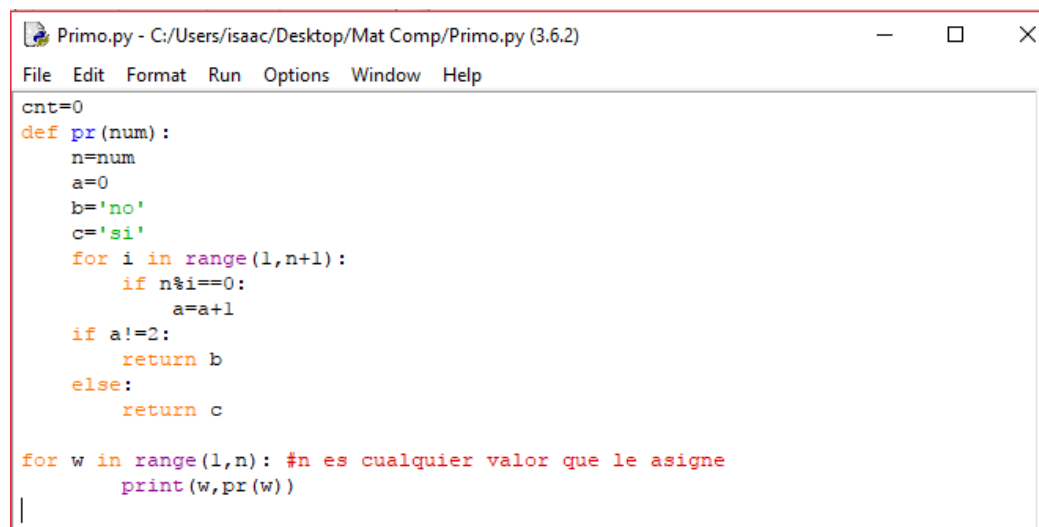
En este reporte se menciona las descripciones que tienen los algoritmos de la sucesión de Fibonacci y los números primos junto con sus graficas de comparación.

Número primo

Primero partimos con su definición, Número entero que solamente es divisible por él mismo (positivo y negativo) y por la unidad (positiva y negativa).

"el 2 es un número primo porque solamente es divisible por 2, -2, 1 y -1"

Con el siguiente algoritmo se puede identificar si un número es primo o no mediante la definición que ya tenemos de este número.



```
Primo.py - C:/Users/isaac/Desktop/Mat Comp/Primo.py (3.6.2)
File Edit Format Run Options Window Help
cnt=0
def pr(num):
    n=num
    a=0
    b='no'
    c='si'
    for i in range(1,n+1):
        if n%i==0:
            a=a+1
    if a!=2:
        return b
    else:
        return c

for w in range(1,n): #n es cualquier valor que le asigne
    print(w,pr(w))
```

Su complejidad es de $O(n)$

Como conclusión se puede observar que el número primo es el que se puede dividir solamente entre dos números: en sí mismo y el uno, como habíamos aclarado en la definición.

Fibonacci

Igual que con el primo, definimos cual es la sucesión de Fibonacci, se trata de una sucesión infinita de números naturales que comienza con los números 1 y 1, y a partir de ellos, cada término se obtiene sumando los dos anteriores:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1.597...

A los elementos de esta sucesión se les llama números de Fibonacci. El nombre de sucesión de Fibonacci se lo debe a Leonardo de Pisa, matemático italiano del siglo XIII también conocido como Fibonacci.

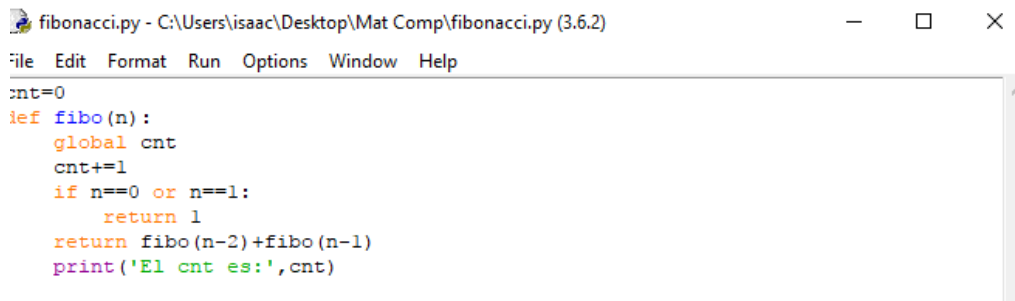
Esta sucesión no tendría nada de particular sino fuera porque aparece repetidamente en la naturaleza y, además, tiene numerosas aplicaciones en ciencias de la computación, matemáticas y teoría de juegos, entre otras.

Por medio de tres algoritmos distintos se puede obtener la sucesión de Fibonacci que se mencionaran a continuación.

Recursivo

Su función es repetir el ciclo hasta obtener el número de Fibonacci en su posición deseada.

Código:

A screenshot of a Python script editor window titled 'fibonacci.py - C:\Users\isaac\Desktop\Mat Comp\fibonacci.py (3.6.2)'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is as follows:

```
cnt=0
def fibo(n):
    global cnt
    cnt+=1
    if n==0 or n==1:
        return 1
    return fibo(n-2)+fibo(n-1)
print('El cnt es:',cnt)
```

Iterativo

En este algoritmo intercambia los datos de las variables asignando valores anteriores a dos variables temporales que su fin es obtener, igual que en el anterior algoritmo, un número de Fibonacci.

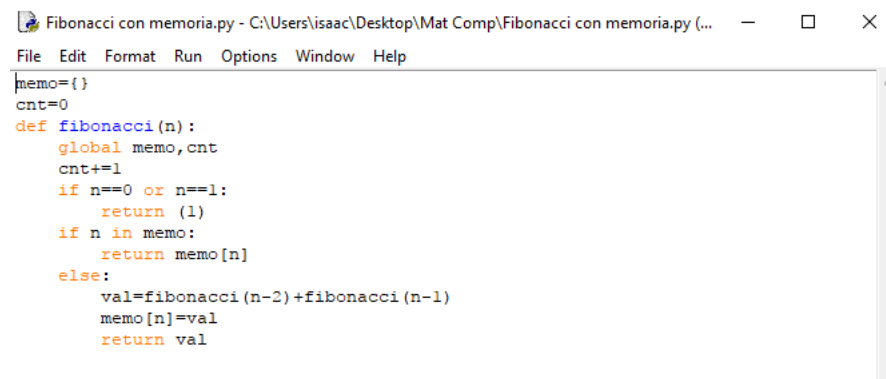
Código:

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

Su complejidad del algoritmo es de $O(n-2)$; $n > 3$

Con memoria

Por último, tenemos el algoritmo con memoria que básicamente declara un arreglo en donde se encuentran los números en su respectiva posición, esto es para tener guardada esta información por si se llegara requerir el número al momento de ejecutar.



```
Fibonacci con memoria.py - C:\Users\isaac\Desktop\Mat Comp\Fibonacci con memoria.py (...)
```

```
File Edit Format Run Options Window Help
memo={}
cnt=0
def fibonacci(n):
    global memo, cnt
    cnt+=1
    if n==0 or n==1:
        return (1)
    if n in memo:
        return memo[n]
    else:
        val=fibonacci(n-2)+fibonacci(n-1)
        memo[n]=val
        return val
```

Con este código se logró capturar los datos necesarios para las gráficas

```
For i in range(2,200):
```

```
    cnt=0
```

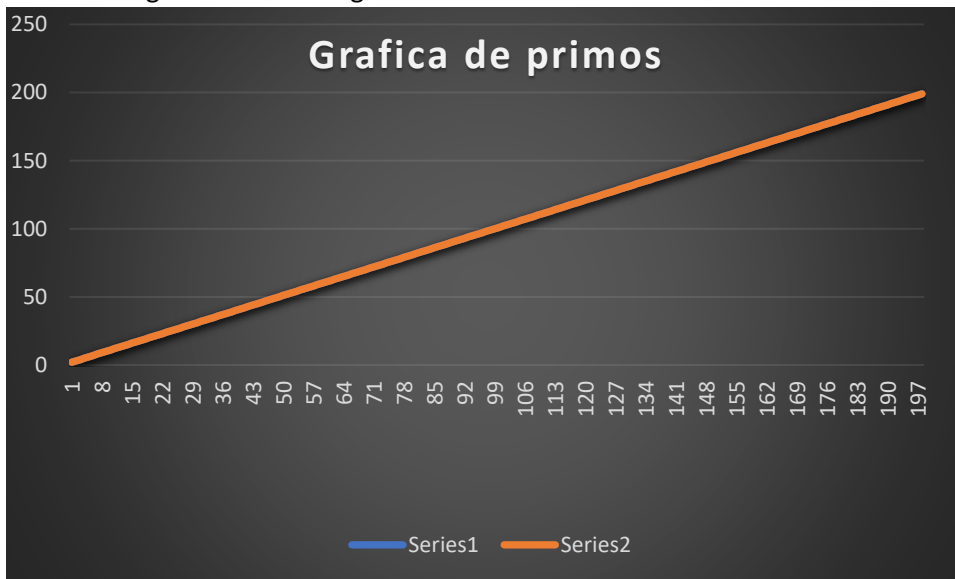
```
    print(i primo(i), cnt)
```

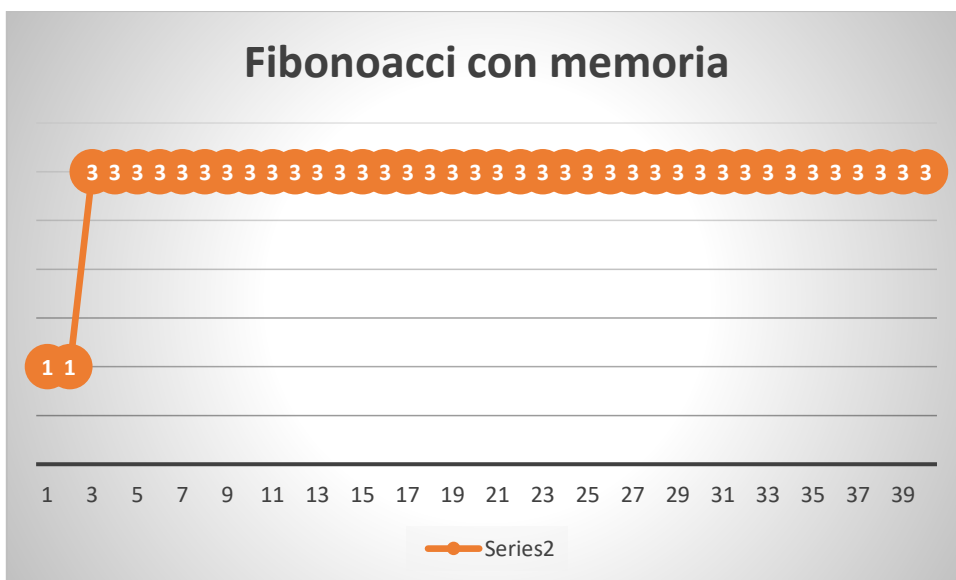
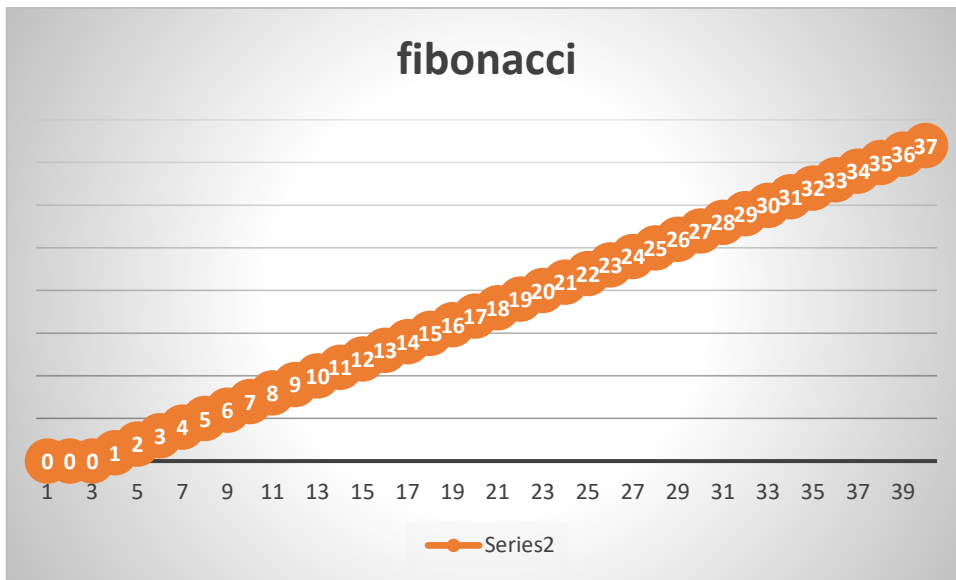
```
for w in range (0,40):
```

```
    cnt=0
```

```
    print(w+1, fibonacci(w), cnt) #fibonacci o fiboniacci, dependiendo de cual se usó
```

Las gráficas son las siguientes:





Como conclusión...

En cuanto los primos, nos dimos cuenta en la práctica que su complejidad es lineal, ya que se requiere de las operaciones el enésimo término. Además, que se pudo dar un método para obtener la tarea de encontrar si es número primo o no y que se puede modificar el método haciéndolo más sencillo. Y en Fibonacci observamos que todos los algoritmos cumplen el objetivo de obtener el número de Fibonacci que deseamos encontrar mediante sus diferentes métodos.

Con los contadores se pudo demostrar gracias a las gráficas la eficacia que tiene cada uno de los algoritmos y observar cuál era el mejor. Los resultados muestran que “con memoria” si se corre de manera ascendente, los valores se obtenían los valores de un sucesor y se puede notar diferencia con los demás. La ventaja que tiene es que al volver a requerir el término enésimo, no se vuelve a realizar n sumas ya que tiene almacenado el número y lo realiza sin realizar ninguna operación.