Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico Matemáticas

Alumna: **Mayra Alejandra Rivera Rodríguez**

Matricula:**1742899**

Matemáticas Computacionales

Reporte de primo y Fibonacci

En este reporte se expondrá de manera breve lo que es un numero primo y a su vez se dará un código que identifica cuál de los números naturales es primo y cual no. Además de eso también se explicará un poco acerca de la sucesión e Fibonacci y se mostraran unos códigos comparando su precisión de operaciones para calcular dicha sucesión.

¿Qué es un numero primo?

Un numero primo es un numero natural ‘n’ mayor a uno, que tiene únicamente dos divisores distintos, que son el mismo número ‘n’ y el número uno.

Programa en Python: La función del programa que se mostrara a continuación es dado un numero n mayor a uno identificar si es primo.

```
contador=0
def primo (n):
    global contador
    contador=0
    for i in range(2, round(n**(1/2)+1)):
        contador+=1
        if n%i==0:
            return("no es primo")
    return ("es primo")
```

En el peor de los casos este código realiza n^½ operaciones

La siguiente grafica muestra el comportamiento de las operaciones del programa.

Sucesión de Fibonacci

¿Qué es?

Se trata de una sucesión infinita de números naturales que comienza con los números 1 y 1, y a partir de ellos, se obtienen los siguientes términos sumando los dos anteriores.

A continuación, se mostraran 3 códigos que realizan l algoritmo para la sucesión e Fibonacci:

El primer código es un Fibonacci-recursivo:

```
contador=0
def fibo1(n):
    global contador
```

```
    contador+=1
    if n==0 or n==1:
        return(1)
    return(fibo1(n-2)+fibo1(n-1))
```

Lo que hace el código anterior es que al ingresar un numero si es el uno o el cero regresara el numero uno y si no realizara la operación indicada para efectuar el algoritmo.

El segundo código es un Fibonacci-iterativo:

```
contador=0
def fibo2(n):
    va=0
    vp=1
    vap=1
    global contador
    if(n==0 or n==1):
        return(1)
    for i in range (2,n+1):
        contador+=1
        va=vp+vap
        vap=vp
        vp=va
    return va
```

Lo que hace el código anterior es que al ingresar un numero si es un uno o un cero devuelve un uno, pero si no, aplicara un ciclo que efectuará el algoritmo de para encontrar los términos de la sucesión de Fibonacci.

El tercer código es un Fibonacci con memoria:

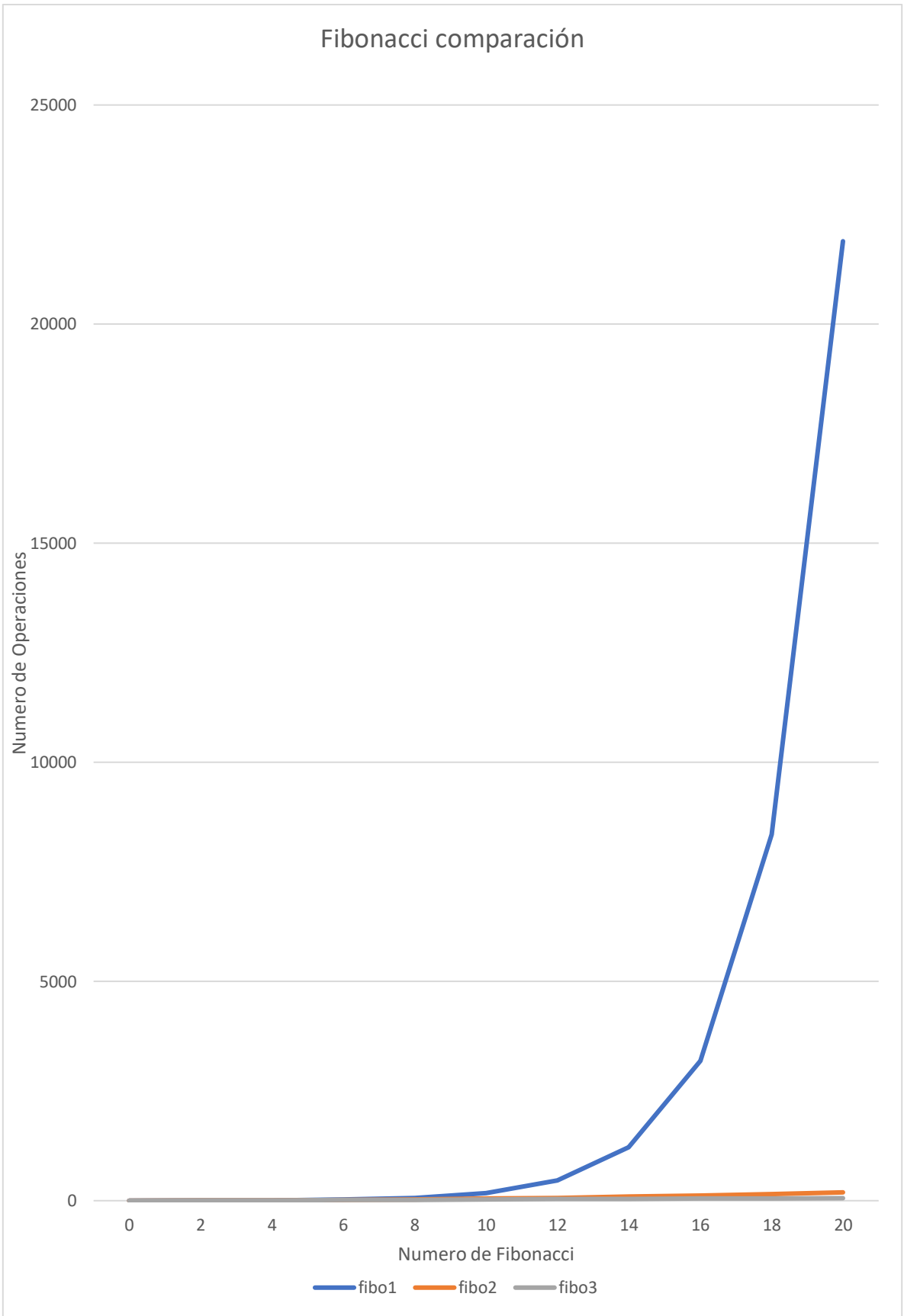
```
memo={}
```

```
contador=0
def fibo3(n):
    global memo
    global contador
    contador+=1
    if n==0 or n==1:
        return(1)
    if n in memo:
        return memo[n]
    else:
        val=fibo3(n-2)+fibo3(n-1)
        memo[n]=val
        return val
```

Este código es muy parecido al Fibonacci recursivo la única diferencia que se puede encontrar entre ellos es que en el Fibonacci con memoria como su nombre lo dice tiene un espacio de memoria para ir guardando los números ya encontrados y esto nos ayuda mas que nada a disminuir el número de operaciones.

A Continuación, se mostrara una tabla con valores obtenidos al correr el algoritmo; además de una tabla comparativa del numero de operaciones que realiza cada uno de los tres para así encontrar cual es el más eficiente.

	fibo1	fibo2	fibo3
0	1	0	1
2	3	1	4
4	9	6	10
6	25	15	16
8	67	28	22
10	177	45	28
12	465	66	34
14	1219	91	40
16	3193	120	46
18	8361	153	52
20	21891	190	58


Fibonacci comparación
25000
20000
15000
10000
5000
0
Numero de Operaciones
0
2
4
6
8
10
12
14
16
18
20
Numero de Fibonacci
fibo1
fibo2
fibo3

Y al observar la tabla de valores y la tabla comparativa nos damos cuenta de que el más eficiente de los tres el Fibonacci con memoria ya que con cada número que valla guardando disminuye el numero de operaciones a realizar.

Conclusiones

El código para el numero primo es muy eficiente ya que tiene un margen de error no muy elevado.

Y para lo de Fibonacci lo que podemos concluir es que para mayor eficiencia hay que usar el tercer código o bien el Fibonacci con memoria, aun que claro los tres códigos funcionan correctamente para encontrar dicha sucesión ya antes mencionada.