

# MATEMATICAS COMPUTACIONALES

## Evaluación no.4

Leslie Jaressy Morales Ortiz. 1661971

### Primos

Ahora bien, pero. . . ¿qué es un número primo? La definición clásica o escolar que todos conocemos es algo así:

**Definición 1.1** (Número primo). Sea  $n \in \mathbb{N}$ .  $n$  es primo sí y sólo sí los únicos divisores que posee son 1 y  $n$ . Esta definición se generaliza para los dominios de integridad, lo cual da lugar al concepto de elemento primo.

**Definición 1.2** (Elemento primo). Sea  $D$  un dominio de integridad y sea  $n \in D$ .  $n$  es un elemento primo si y solo si:

- a)  $n$  diferente de 0
- b)  $n$  no es una unidad.
- c) Si  $p$  divide a  $ab$  con  $a, b \in D$  entonces  $p|a$  o bien  $p|b$ .

Nos enfrentamos ahora al problema en cuestión, ¿cómo certificar que un número es primo? Si se le pregunta a un escolar lo más probable es que te conteste que miremos la tabla de número primos que tiene en su libro de textos (si es muy vago. . . o muy listo). O bien podría contestar que probemos a dividir por todos los números menores que él (ahora bien, si es realmente listo dirá que solo hace falta hasta su raíz cuadrada). Estas ideas simples fueron realmente los primeros test de primalidad. El primero es conocido como la criba de Eratóstenes y es debido a Eratóstenes (siglo II a.C.)

### **Código.**

El presente código tiene la función de mostrar el número de operaciones que se realizar en los elementos de un arreglo declarado el cual opera las condiciones de que sea un numero primo o no, además se especifica que debe considerar un salto de 50 elementos para reducir su totalidad, esto solo como fines prácticos una mejor apreciación de la gráfica.

```
def primo(n):
    cnt=0
    for i in 2, (n**0.5):
        cnt=cnt+1
        if ((n%i)==0):
            #return("no es primo")
            break
        #return("si es primo")
    return cnt

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

for w in range(1,10000,50):
    print(w,primo(w))
|
```

### **Tabla.**

Elemento	Operaciones
1	0
51	2
101	8
151	10
201	2
251	14
301	6
351	2
401	18

451	10
501	2
551	18
601	23
651	2
701	24
751	25
801	2
851	22
901	16
951	2
1001	6
1051	30
1101	2
1151	32
1201	33
1251	2
1301	34
1351	6
1401	2
1451	36
1501	18
1551	2
1601	38
1651	12
1701	2
1751	16
1801	40
1851	2
1901	42

1951	42
2001	2
2051	6
2101	10
2151	2
2201	30
2251	45
2301	2
2351	46
2401	6
2451	2
2501	40
2551	49
2601	2
2651	10
2701	36
2751	2
2801	51
2851	51
2901	2
2951	12
3001	53
3051	2
3101	6
3151	22
3201	2
3251	55
3301	55
3351	2
3401	18

3451	6
3501	2
3551	52
3601	12
3651	2
3701	59
3751	10
3801	2
3851	60
3901	46
3951	2
4001	61
4051	62
4101	2
4151	6
4201	63
4251	2
4301	10
4351	18
4401	2
4451	65
4501	6
4551	2
4601	42
4651	66
4701	2
4751	67
4801	67
4851	2
4901	12

4951	68
5001	2
5051	69
5101	69
5151	2
5201	6
5251	58
5301	2
5351	71
5401	10
5451	2
5501	72
5551	6
5601	2
5651	73
5701	74
5751	2
5801	74
5851	74
5901	2
5951	10
6001	16
6051	2
6101	76
6151	76
6201	2
6251	6
6301	77
6351	2
6401	36

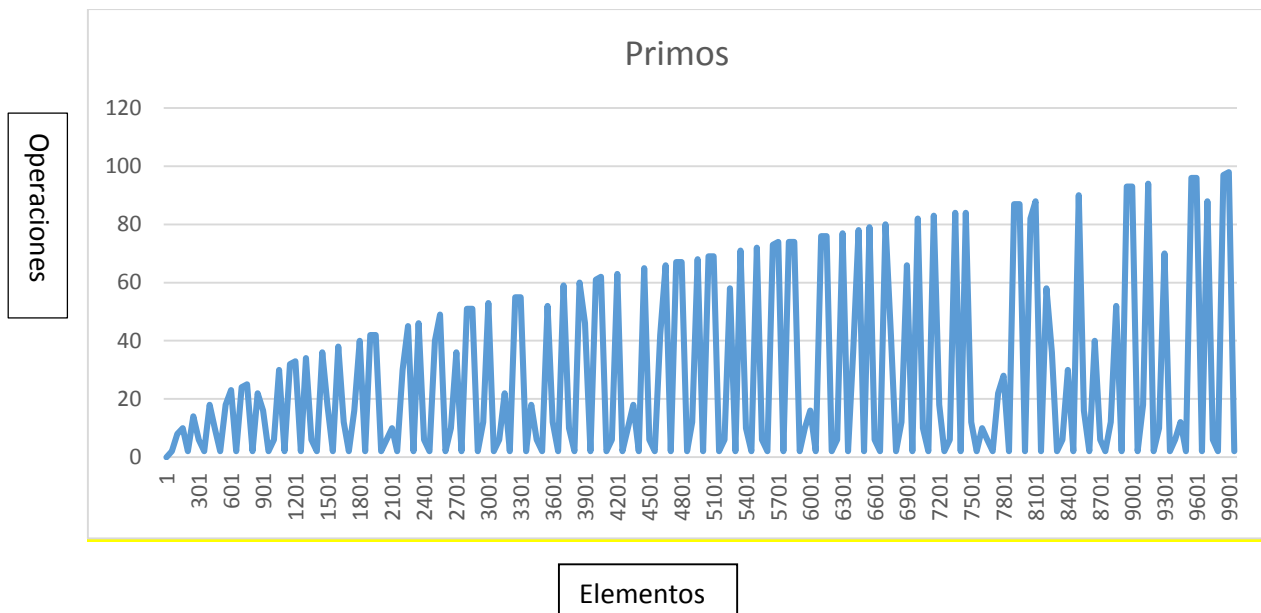
6451	78
6501	2
6551	79
6601	6
6651	2
6701	80
6751	42
6801	2
6851	12
6901	66
6951	2
7001	82
7051	10
7101	2
7151	83
7201	18
7251	2
7301	6
7351	84
7401	2
7451	84
7501	12
7551	2
7601	10
7651	6
7701	2
7751	22
7801	28
7851	2
7901	87

7951	87
8001	2
8051	82
8101	88
8151	2
8201	58
8251	36
8301	2
8351	6
8401	30
8451	2
8501	90
8551	16
8601	2
8651	40
8701	6
8751	2
8801	12
8851	52
8901	2
8951	93
9001	93
9051	2
9101	18
9151	94
9201	2
9251	10
9301	70
9351	2
9401	6



9451	12
9501	2
9551	96
9601	96
9651	2
9701	88
9751	6
9801	2
9851	97
9901	98
9951	2

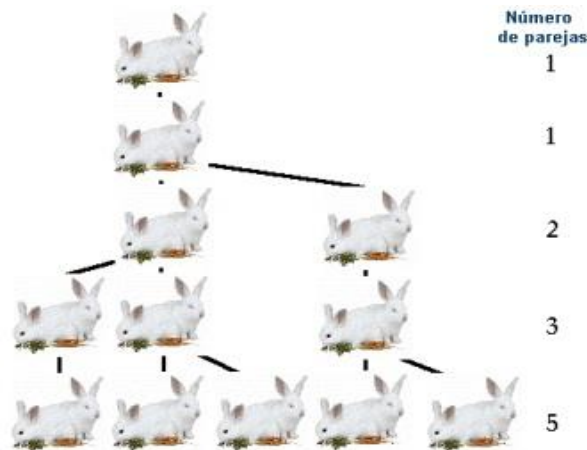
### **Grafica**



En la presente grafica se puede visualizar el crecimiento y decrecimiento para los elementos del arreglo.

## La Sucesión de Fibonacci

La **sucesión de Fibonacci**, en ocasiones también conocida como secuencia de Fibonacci o incorrectamente como serie de Fibonacci, es en sí una sucesión matemática infinita. Consta de una serie de números naturales que se suman de a 2, a partir de 0 y 1. Básicamente, la sucesión de Fibonacci se realiza sumando siempre los últimos 2 números.



En este gráfico vemos que el número de parejas a lo largo de los meses coincide con los términos de la sucesión.

Veamos con detalle estos números. **1; 1; 2; 3; 5; 8; 13; 21; 34; 55; 89; 144....**

Es fácil ver que cada término es la suma de los dos anteriores. Pero existe entre ellos otra relación curiosa, el cociente entre cada término y el anterior se va acercando cada vez más a un número muy especial, ya conocido por los griegos y

aplicado en sus esculturas y sus templos: el número áureo. 
$$\Phi = \frac{1 + \sqrt{5}}{2} = 1.618039...$$

## Algoritmo recursivo

### código

```
ley = {0: 0, 1: 1}#Declaracion de los primeros elementos
def fib(x):
    if x not in ley:#Proceso
        ley[x] = fib(x- 1) + fib(x - 2)
    return ley[x]
for w in range(1,51):#Rango a valorar
    print(w,fib(w))#Imprime valor de ( x) y su posicion
```

En esta sección trabajamos con Fibonacci, donde a partir de un algoritmo recursivo declaramos un arreglo (1,51) donde tuvimos que valorar la secuencia de dicha sucesión para posteriormente que el programa ejecute y muestre la posición correspondiente al número que ingrese.

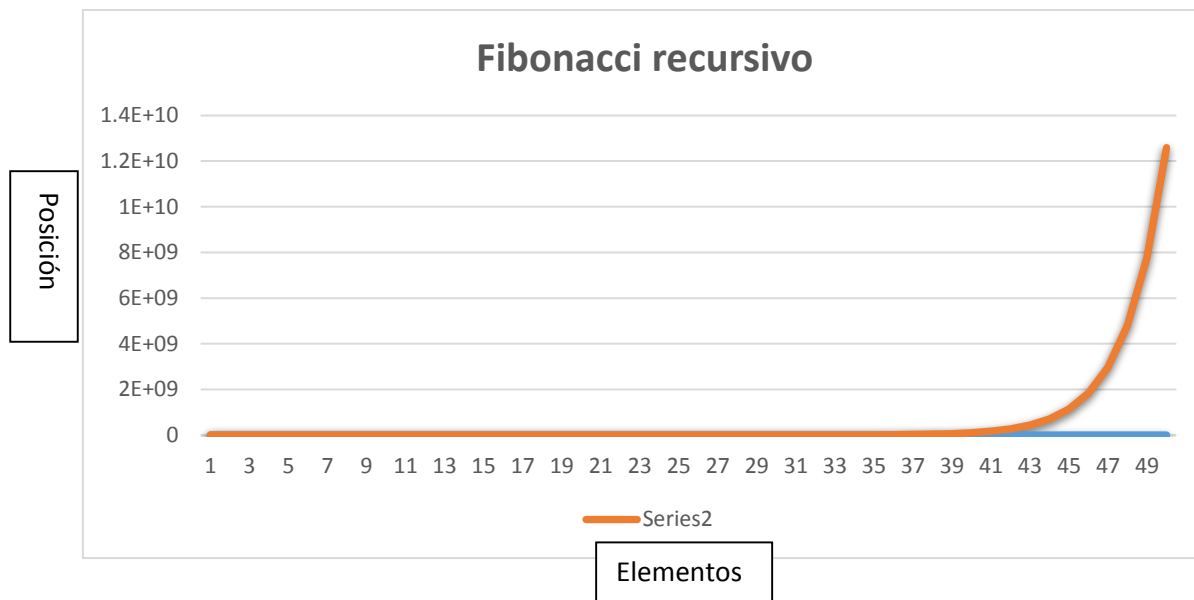
### Tabla

Elementos	Posición
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55
11	89
12	144
13	233
14	377
15	610

16	987
17	1597
18	2584
19	4181
20	6765
21	10946
22	17711
23	28657
24	46368
25	75025
26	121393
27	196418
28	317811
29	514229
30	832040
31	1346269
32	2178309
33	3524578
34	5702887
35	9227465
36	14930352
37	24157817
38	39088169
39	63245986
40	102334155
41	165580141
42	267914296
43	433494437
44	701408733
45	1134903170

46	1836311903
47	2971215073
48	4807526976
49	7778742049
50	12586269025

## Grafica



## Algoritmo iterativo

### código

```
def fiboiter(n):
    global cnt
    fib=[1,1]
    for k in range(2,n+1):
        cnt+=1
        fib.append(fib[k-1]+fib[k-2])
    return fib[n]

for n in range(0,101):
    cnt=0
    a=fiboiter(n)
    cntr,cnt=cnt,0
    print(n,fiboiter(n))
```

La tercera sección se trata de aplicar Fibonacci interactivo, de esta forma tratamos de mejorar el rendimiento logrando que la función solo calcule una vez cada termino

 **tabla**

Elementos	Posición
0	1
1	1
2	2
3	3
4	5
5	8
6	13
7	21
8	34
9	55
10	89
11	144
12	233
13	377
14	610
15	987
16	1597
17	2584
18	4181
19	6765
20	10946
21	17711
22	28657
23	46368
24	75025

25	121393
26	196418
27	317811
28	514229
29	832040
30	1346269
31	2178309
32	3524578
33	5702887
34	9227465
35	14930352
36	24157817
37	39088169
38	63245986
39	102334155
40	165580141
41	267914296
42	433494437
43	701408733
44	1134903170
45	1836311903
46	2971215073
47	4807526976
48	7778742049
49	1,2586E+10
50	2,0365E+10
51	3,2951E+10
52	5,3316E+10
53	8,6268E+10
54	1,3958E+11

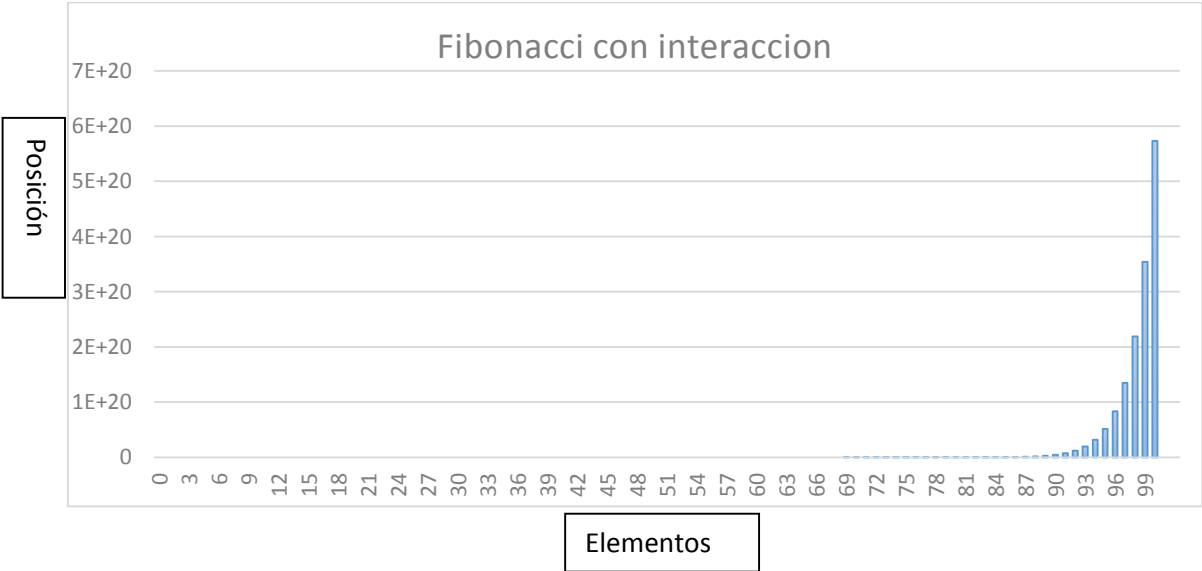
55	2,2585E+11
56	3,6544E+11
57	5,9129E+11
58	9,5672E+11
59	1,548E+12
60	2,5047E+12
61	4,0527E+12
62	6,5575E+12
63	1,061E+13
64	1,7168E+13
65	2,7778E+13
66	4,4946E+13
67	7,2723E+13
68	1,1767E+14
69	1,9039E+14
70	3,0806E+14
71	4,9845E+14
72	8,0652E+14
73	1,305E+15
74	2,1115E+15
75	3,4165E+15
76	5,5279E+15
77	8,9444E+15
78	1,4472E+16
79	2,3417E+16
80	3,7889E+16
81	6,1306E+16
82	9,9195E+16
83	1,605E+17
84	2,597E+17



85	4,202E+17
86	6,7989E+17
87	1,1001E+18
88	1,78E+18
89	2,8801E+18
90	4,66E+18
91	7,5401E+18
92	1,22E+19
93	1,974E+19
94	3,194E+19
95	5,1681E+19
96	8,3621E+19
97	1,353E+20
98	2,1892E+20
99	3,5422E+20
100	5,7315E+20



Grafica



## Algoritmo de recursión con memoria

### Código

```
global contador
contador = 0
cache = {0: 0, 1: 1}

def fib(m):
    global contador
    contador = contador + 1
    if m not in cache:
        cache[m] = fib(m - 1) + fib(m - 2)
    return cache[m]

y=[]
e=[]
for i in range(1,100):
    y.append(i)
    contador=0
    cache = {0: 0, 1: 1}
    fib(i)
    e.append(contador)
```

Anteriormente estuvimos trabajando con un algoritmo recursivo, éste en particular también pertenece a la misma clase con una pequeña diferencia de que contiene una memoria donde guarda las operaciones ya hechas, de esta forma si se ingresa un elemento y arroja la posición con respecto a Fibonacci y después se vuelve a solicitar la posición del elemento ya mencionado no demorara en mostrar debido a que las operaciones ya fueron realizadas y será instantáneo porque se gurdo en memoria

### Tabla

Elemento	Posición
1	1
2	3
3	5
4	7
5	9
6	11

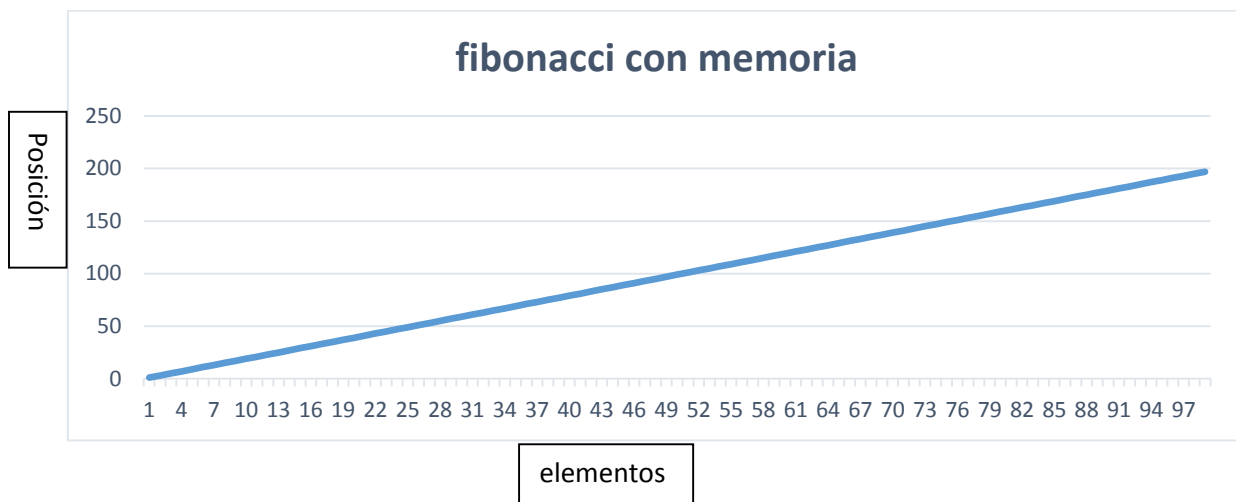
7	13
8	15
9	17
10	19
11	21
12	23
13	25
14	27
15	29
16	31
17	33
18	35
19	37
20	39
21	41
22	43
23	45
24	47
25	49
26	51
27	53
28	55
29	57
30	59
31	61
32	63
33	65
34	67
35	69
36	71

37	73
38	75
39	77
40	79
41	81
42	83
43	85
44	87
45	89
46	91
47	93
48	95
49	97
50	99
51	101
52	103
53	105
54	107
55	109
56	111
57	113
58	115
59	117
60	119
61	121
62	123
63	125
64	127
65	129
66	131

67	133
68	135
69	137
70	139
71	141
72	143
73	145
74	147
75	149
76	151
77	153
78	155
79	157
80	159
81	161
82	163
83	165
84	167
85	169
86	171
87	173
88	175
89	177
90	179
91	181
92	183
93	185
94	187
95	189
96	191

97	193
98	195
99	197

### Grafica



### Conclusión.

A principios de esta evaluación trabajamos con los números primos, fue de aquí donde partimos para la elaboración de un programa en el cual tiene como funcionalidad mostrar el número de operaciones que se realizan en un elemento, a partir de esto podemos determinar si es primo o no, de hecho, el programa que use en esta sección puede arrojar inmediatamente si un número es primo o no, pero debido a que esto no fue lo requerido por el docente lo deje comentado.

Así mismo la elaboración del segundo trabajo se dividió en tres partes desglosadas de la sucesión Fibonacci, definimos a la sucesión y sus características, para la segunda actividad pusimos en práctica el algoritmo de recursión cuyo proceso en lo personal me parece que fue el más sencillo, posteriormente para la actividad 3 se aplicó el algoritmo interactivo el cual permite que se realice solo una operación mejorando la eficiencia del programa principal ;por ultimo cerramos esta evaluación

con el algoritmo de recursión con memoria , el cual me parece que podría ser de los más eficientes pues eso de tomar en cuenta un dato que sea el encargado de guardar las operaciones que ya se hicieron es muy favorable , esto orientado a que en la actualidad se buscan productos eficientes , prácticos , rápidos para que cualquier tipo de usuario pueda poner en practica .

Por otra parte, resalto que tuve algunas complicaciones con terminar a tiempo el proyecto, esto debido a que no me informe bien de las actividades solicitados, aunque he de confesar que me quedo con un mejor conocimiento del uso de Excel el cual fue de suma importancia para la plasmación de los datos finales y las representaciones gráficas.