Reporte de test para números primos y la sucesión de Fibonacci.

María Luisa Borrego Riojas
Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas
Matemáticas Computacionales

En este reporte se buscará definir y analizar los algoritmos con relación a los números primos, así como también a los números que conforman la sucesión de Fibonacci (usando los algoritmos de Fibonacci recursivo, iterativo y recursivo con memoria). El objetivo es poder usar cada uno de ellos en la plataforma de Python.

1. Test para números primos.

Gracias a su naturaleza, los números primos han sido un tema de gran interés a través del tiempo, o con mayor exactitud, el cómo obtenerlos. Matemáticos importantes como lo son Eratóstenes, Euclides, Mersenne y Euler dedicaron gran parte de sus investigaciones a ellos.

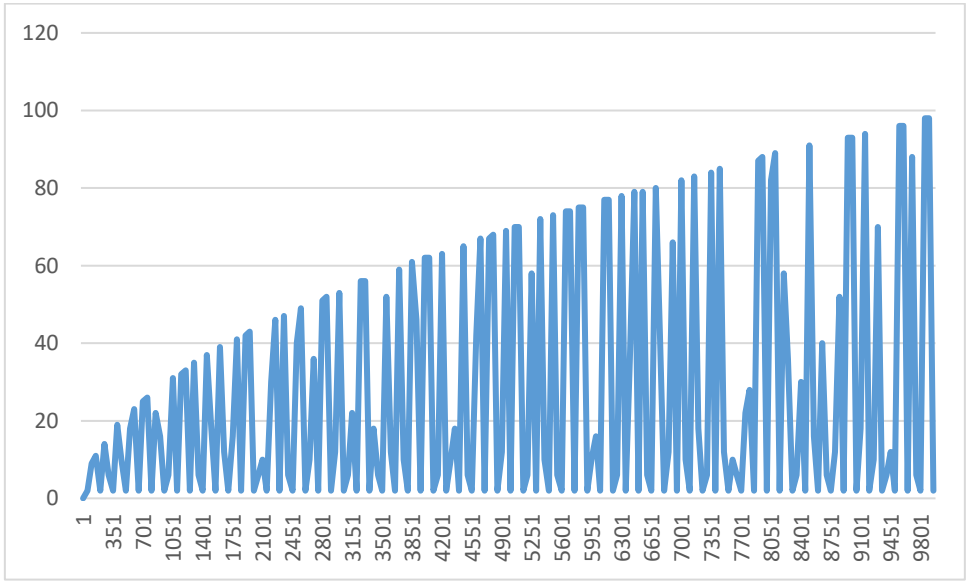
Para poder analizar si un número es o no primo, debemos tener claro primeramente lo que es un número primo. Éstos son aquellos números que sólo son divisibles por ellos mismos y por la unidad. Esto significa que todo número primo tiene únicamente 2 divisores.

Entonces, para saber si un número es primo, lo dividimos sucesivamente por los primeros números primos (2,3,5,7,11,…) hasta que encontremos una división exacta (en cuyo caso, el número no sería primo) o hasta que el cociente sea menor que el divisor (el número es primo).

Esto en un formato para Python sería algo así:

```
def Primo(a):
	if a<=1:
		return NoPrimo
	b=2
	while(b*b<=a):
		if a%b==0:
			return NoPrimo
		m+=1
	return Primo
```

En el peor de los casos, el algoritmo anterior realizaría $\sqrt{n}$ operaciones, usualmente cuando el número si es primo. A continuación, la gráfica mostrando la cantidad de operaciones del algoritmo para los números desde 1 a 10,000:



2. Sucesión de Fibonacci.

Fibonacci es uno de los nombres más famosos en matemáticas, aunque realmente su nombre era Leonardo de Pisa, debido a una mala traducción ahora todo el mundo lo conoce como Fibonacci. La manera en la que este hombre fue inmortalizado, fue a

través de su famosa secuencia {0, 1, 1, 2, 3, 5, 8, 13, …} para sorpresa de Leonardo, pues esta secuencia era sólo un pequeño problema de conejos dentro de su libro “Liber Abaci”.

Cada número en la secuencia Fibonacci (empezando por el 2) es la suma de los dos números que le preceden.

Esto es:

= 1, 1, (1+1), (2+1), (2+3), (3+5), …

= 1, 1, 2 , 3 , 5 , 8 , …

Si transformamos esto en fórmula sería:

$F_{n+1} = F_{n-1} + F_n$; para $n \geq 3$

donde $F_1 = 1$ y $F_2 = 1$

En teoría, la fórmula continúa funcionando con todos los números de ahí hasta infinito.

A continuación, se mostrarán 3 códigos diferentes relacionados con los números Fibonacci. En primer lugar, tenemos la función recursiva, en donde dado un número n se pueda calcular el n-ésimo número de la sucesión de manera recursiva y cuenta con una complejidad de $O(f_n)$. Esto, en Python es:

```
def FiboRecursivo(n):
    if n==1 or n==2:
        return 1
    return FiboRecursivo(n-1) + FiboRecursivo(n-2)
```

Lamentablemente, la función previa repite muchas operaciones, por lo que no es muy eficaz. Para mejorarlo, se puede optimizar creando un arreglo para que cada término sea calculado una sola vez. A esto le llamaremos la función Fibonacci iterativo, con una complejidad de $O(n)$ y su pseudocódigo es:

```
def FiboIterativo(n):
      f=[1,1]
      for i in range (2, n+1):
            f.append (f[i-1] + f[i-2])
      return f[n]
```

A diferencia de la función recursiva, la iterativa tiene una cantidad de operaciones mucho menor, aunque sigue contando con desventajas al momento de compararlo con otros algoritmos.

Por último, tenemos la función Fibonacci recursivo con memoria, tiene la misma complejidad que el iterativo de $O(n)$ y funciona muy similar al recursivo, pero al contar con memoria no calcula valores de más. El pseudocódigo es:

```
A = {}
def Fibo3(n):
      global A
      if n==0 or n==1:
            return 1
      if n in A:
            return A[n]
      else:
            A[n] = Fibo3(n-2) + Fibo3(n-1)
            return A[n]
```

Conclusiones.

En el reporte se pudo mostrar exitosamente que es posible crear un método para identificar si un número es primo o no, y después de esto el algoritmo creado puede ser modificado para mejorar su eficacia. Esto puede ser de gran utilidad en investigaciones en un futuro, pues es más rápido que el método tradicional con hoja y papel.

También, se trabajó con la serie de Fibonacci de 3 maneras distintas, buscando la mejor manera de encontrar la sucesión para que se realice la operación de una manera óptima y eficaz. Dado que los números Fibonacci tienen innumerables aplicaciones con la vida, no será muy difícil encontrar maneras de usarlo en trabajos, problemas, o situaciones cotidianas.