	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Jesus Cruz Navarro
Asignatura:	Estructura de Datos y Algoritmos II
Grupo:	Grupo 1
No de Práctica(s):	Práctica 0
Integrante(s):	Edwin Jaret Santiago Díaz
No. de Equipo de cómputo empleado:	
No. de Lista o Brigada:	
Semestre:	2022 - 2
Fecha de entrega:	6 Febrero 2022
Observaciones:	

CALIFICACIÓN: _____

Objetivos

1.- Que el alumno utilice el IDE Visual Studio Code con el lenguaje de programación Python.
2.- Que el alumno observe la diferencia en términos de complejidad asintótica de diferentes algoritmos para resolver un mismo problema.

Desarrollo

Se creó un programa para calcular el tiempo de ejecución del método recursivo e iterativo en la secuencia de Fibonacci desde el número 0 hasta el deseado y a la vez, se imprimirá cada número de esta secuencia con respecto a su tiempo de ejecución. Al final de la ejecución de cada método se hará uso de la librería **matplotlib** para graficar los datos arrojados.

Para hacer uso de la librería **matplotlib** se requiere de un entorno virtual, en este caso no puede ser incluido en el archivo zip debido a su tamaño y cantidad de archivos.

En ambos métodos se calculó el número **40** de la sucesión de Fibonacci.

Fibonacci Recursivo

- La función recibe como parámetro el número deseado
- Se conocen que los primeros 2 números son 1.
- Para calcular el número **n** el algoritmo se llama así mismo 2 veces pasándole como parámetro los 2 números anteriores a este hasta llegar a los números 1 o 0.
- Al llegar el numero deseado, la función retornará el resultado de este.

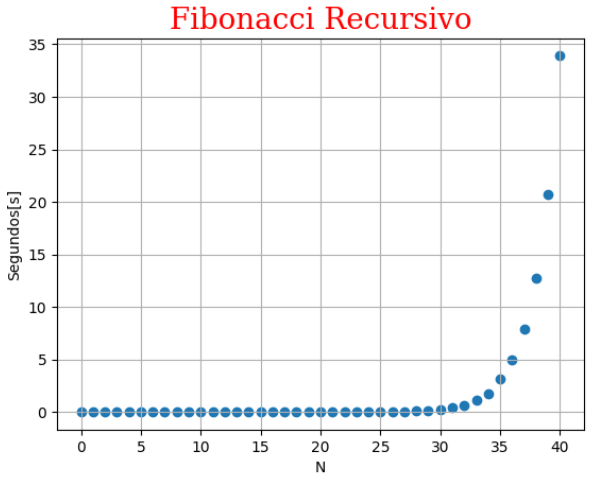
```python
# Fibonacci recursivo
def fibo_recursivo(n):
    # Se conoce los 2 primeros numeros de la sucesion de Fibonacci
    f0 = 0 #* O(1)
    f1 = 1 #* O(1)
    if n == 0 or n == 1:
        return 1
    # Para calcular el n numero, se vuelve a llamar a la funcion 2 veces pasandole
    # los 2 valores anterios de dicho numero
    # Al final, se retornará el numero de la posicion n de Fibonacci
    return fibo_recursivo(n-1) + fibo_recursivo(n-2) #* O(n^c)

#* 2 + n^c = O(2 + n^c) = O(n^c)
#* Complejidad exponencial O(n^c)
```

El resultado de Fibonacci de 40 es: **102334155**

La complejidad de este método es exponencial debido a que se realiza dos veces recursividad en una misma línea, generando que la computadora trabaje mucho.

En el Fibonacci Recursivo se obtuvieron los siguientes resultados al imprimir cada número de la sucesión con respecto a su tiempo de ejecución.

```
**Fibonacci Recursivo**
El numero 0 de Fibo es 1 y tardó: 1.1003576219081879e-06[s]
El numero 1 de Fibo es 1 y tardó: 9.997747838497162e-07[s]
El numero 2 de Fibo es 2 y tardó: 1.600012183189392e-06[s]
El numero 3 de Fibo es 3 y tardó: 2.1997839212417603e-06[s]
El numero 4 de Fibo es 5 y tardó: 2.9997900128364563e-06[s]
El numero 5 de Fibo es 8 y tardó: 2.400018274784088e-06[s]
El numero 6 de Fibo es 13 y tardó: 4.500150680541992e-06[s]
El numero 7 de Fibo es 21 y tardó: 5.600042641162872e-06[s]
El numero 8 de Fibo es 34 y tardó: 8.599832653999329e-06[s]
El numero 9 de Fibo es 55 y tardó: 1.2699980288743973e-05[s]
El numero 10 de Fibo es 89 y tardó: 2.079969272017479e-05[s]
El numero 11 de Fibo es 144 y tardó: 2.9799994081258774e-05[s]
El numero 12 de Fibo es 233 y tardó: 7.699988782405853e-05[s]
El numero 13 de Fibo es 377 y tardó: 7.429998368024826e-05[s]
El numero 14 de Fibo es 610 y tardó: 0.00011830031871795654[s]
El numero 15 de Fibo es 987 y tardó: 0.00018990039825439453[s]
El numero 16 de Fibo es 1597 y tardó: 0.0003387997858226299 3[s]
El numero 17 de Fibo es 2584 y tardó: 0.0005598999559879303[s]
El numero 18 de Fibo es 4181 y tardó: 0.0008156001567840576[s]
El numero 19 de Fibo es 6765 y tardó: 0.0013128002174198627[s]
El numero 20 de Fibo es 10946 y tardó: 0.002280899789184332[s]
El numero 21 de Fibo es 17711 y tardó: 0.003702299669384956 4[s]
El numero 22 de Fibo es 28657 y tardó: 0.007198599632829428[s]
El numero 23 de Fibo es 46368 y tardó: 0.01042129984125495[s]
El numero 24 de Fibo es 75025 y tardó: 0.015553600154817104[s]
El numero 25 de Fibo es 121393 y tardó: 0.03517810022458434[s]
El numero 26 de Fibo es 196418 y tardó: 0.047434099949896336[s]
El numero 27 de Fibo es 317811 y tardó: 0.06970890006050467[s]
El numero 28 de Fibo es 514229 y tardó: 0.1140934000723064[s]
El numero 29 de Fibo es 832040 y tardó: 0.1685124998912215 2[s]
El numero 30 de Fibo es 1346269 y tardó: 0.2694296003319323[s]
El numero 31 de Fibo es 2178309 y tardó: 0.4230462000705302[s]
El numero 32 de Fibo es 3524578 y tardó: 0.6932195001281798[s]
El numero 33 de Fibo es 5702887 y tardó: 1.1123581998981535[s]
El numero 34 de Fibo es 9227465 y tardó: 1.795147399883717 3[s]
El numero 35 de Fibo es 14930352 y tardó: 3.1716563003137708[s]
El numero 36 de Fibo es 24157817 y tardó: 4.975308999884874[s]
El numero 37 de Fibo es 39088169 y tardó: 7.891635499894619[s]
El numero 38 de Fibo es 63245986 y tardó: 12.801440800074488[s]
El numero 39 de Fibo es 102334155 y tardó: 20.736298200208694[s]
El numero 40 de Fibo es 165580141 y tardó: 33.89589259959757[s]
```



Después de calcular el número 35, el programa tomó más tiempo y se puede observar que es de complejidad exponencial.

Fibonacci Iterativo:

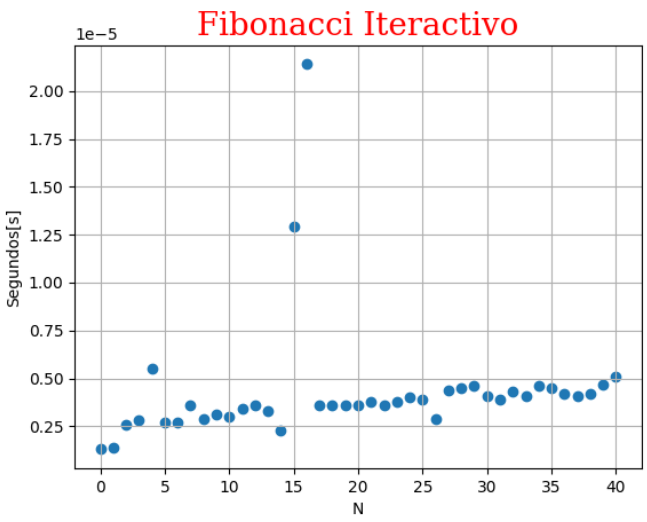
- La función recibe como parámetro el número deseado
- Se conocen que los primeros 2 números son 1.
- Para calcular el número **n** el algoritmo recorre con un **ciclo for** desde el 2 hasta el **n** para así, sumar los 2 números actuales (f0 y f1), después al **f0** se le asigna el valor de **f1** y al **f1** se le asigna el valor de la suma previamente realizada.
- Al llegar el numero deseado, la función retornará el resultado de este.

```python
#Fibonacci Iterativo
def fib_iterativo(n):
    # Se conoce los 2 primeros numeros de la sucesion de Fibonacci
    f0 = 0 #* O(1)
    f1 = 1 #* O(1)
    if n == 0 or n == 1:
        return 1
    # Con un ciclo for se recorre desde la posicion 2 hasta el n+1 numero
    # (ya que un ciclo for es exclusivo)
    for i in range(2, n+1): #* O(n)
        # La suma de los 2 siguientes numeros
        fn = f0 + f1 #* O(1) * n
        # El primer numero se convierte en el segundo numero sumado
        f0 = f1 #* O(1) * n
        # El segundo numero se convierte en el valor de la suma realizada previamente
        f1 = fn #* O(1) * n
    # Se retorna el numero obtenido
    return fn #* O(n)

#* 4 + 3n = O(4 + 3n) = O(n)
#* Complejidad Lineal O(n)
```

El resultado de Fibonacci de 40 es: **102334155**

En el Fibonacci Iterativo se obtuvieron los siguientes resultados al imprimir cada número de la sucesión con respecto a su tiempo de ejecución.

```
**Fibonacci Iterativo**
El numero 0 de Fibo es 1 y tardó: 1.300126314163208e-06[s]
El numero 1 de Fibo es 1 y tardó: 1.3997778296470642e-06[s]
El numero 2 de Fibo es 1 y tardó: 2.5997869670391083e-06[s]
El numero 3 de Fibo es 2 y tardó: 2.800021320581436e-06[s]
El numero 4 de Fibo es 3 y tardó: 5.499925464391708e-06[s]
El numero 5 de Fibo es 5 y tardó: 2.699904143810272 2e-06[s]
El numero 6 de Fibo es 8 y tardó: 2.699904143810272 2e-06[s]
El numero 7 de Fibo es 13 y tardó: 3.600027412176132e-06[s]
El numero 8 de Fibo es 21 y tardó: 2.8996728360652924e-06[s]
El numero 9 de Fibo es 34 y tardó: 3.0999071896076202e-06[s]
El numero 10 de Fibo es 55 y tardó: 2.9997900128364563e-06[s]
El numero 11 de Fibo es 89 y tardó: 3.400258719921112e-06[s]
El numero 12 de Fibo es 144 y tardó: 3.600027412176132e-06[s]
El numero 13 de Fibo es 233 y tardó: 3.300141543149948e-06[s]
El numero 14 de Fibo es 377 y tardó: 2.299901098012924e-06[s]
El numero 15 de Fibo es 610 y tardó: 1.2900214642286 3e-05[s]
El numero 16 de Fibo es 987 y tardó: 2.1399930119514465e-05[s]
El numero 17 de Fibo es 1597 y tardó: 3.600027412176132e-06[s]
El numero 18 de Fibo es 2584 y tardó: 3.600027412176132e-06[s]
El numero 28 de Fibo es 317811 y tardó: 4.499685019254 6844e-06[s]
El numero 29 de Fibo es 514229 y tardó: 4.600267857313156e-06[s]
El numero 30 de Fibo es 832040 y tardó: 4.100147634744644e-06[s]
El numero 31 de Fibo es 1346269 y tardó: 3.899913281202316e-06[s]
El numero 32 de Fibo es 2178309 y tardó: 4.299916326999664e-06[s]
El numero 33 de Fibo es 3524578 y tardó: 4.100147634744644e-06[s]
El numero 34 de Fibo es 5702887 y tardó: 4.600267857313156e-06[s]
El numero 35 de Fibo es 9227465 y tardó: 4.500150680541992e-06[s]
El numero 36 de Fibo es 14930352 y tardó: 4.1997991502285e-06[s]
El numero 37 de Fibo es 24157817 y tardó: 4.0996819734573364e-06[s]
El numero 38 de Fibo es 39088169 y tardó: 4.1997991502285e-06[s]
El numero 39 de Fibo es 63245986 y tardó: 4.699919372797012e-06[s]
El numero 40 de Fibo es 102334155 y tardó: 5.09992241859436e-06[s]
```



La grafica nos demuestra que el método es de complejidad lineal, además, para calcular los números después del 35 no tomó tanto tiempo como en el método recursivo.

¿Cómo se graficó?

Para graficar ambos métodos se necesita dos listas de números para el eje de X (representa los números desde el 0 hasta el **n** **[**una lista para el método recursivo y otra para el iterativo**]**) y dos listas de números para el eje Y (representa el tiempo de ejecución de cada número **[**una lista para el método recursivo y otra para el iterativo**]**) y cada valor es agregado a su respectiva lista después de haber sido calculado.

En el caso del tiempo, este se calcula iniciando un cronómetro, ejecutar el método deseado y detener el cronómetro, al final se obtiene la diferencia cuando se detuvo el cronómetro a cuando se inició el cronómetro.

Por último, se imprimen los datos obtenidos de cada número **n**.

RECURSIVO:

```python
#* Se calcula el fibonacci Recursivo
print("\n**Fibonacci Recursivo**")
for i in range (0, n+1):
    # Se inicializa el cronómetro
    tiempoInicial = perf_counter()
    # Se ejecuta el algoritmo
    fibb_rec = fibo_recursivo(i)
    # Se detiene el cronómetro
    tiempoFinal = perf_counter()
    #Se calcula el tiempo tomado a completar el algoritmo
    tiempo = tiempoFinal - tiempoInicial

    # Se agrega a la lista el numero calculado
    x_recursivo.append(i)
    # Se agrega a la lista el tiempo que tardó el programa en calcularlo
    y_recursivo.append(tiempo)

    # Se imprimen los datos
    print(f'El numero {i} de Fibo es {fibb_rec} y tardó: {tiempo}[s]')
print("\n")
```

ITERATIVO;

```python
#* Se calcula el fibonacci Iterativo
print("**Fibonacci Iterativo**")
for i in range (0, n+1):
    # Se inicializa el cronómetro
    tiempoInicial = perf_counter()
    # Se ejecuta el algoritmo
    fibb_ite = fib_iterativo(i)
    # Se detiene el cronómetro
    tiempoFinal = perf_counter()
    #Se calcula el tiempo tomado a completar el algoritmo
    tiempo = tiempoFinal - tiempoInicial

    # Se agrega a la lista el numero calculado
    x_iteractivo.append(i)
    # Se agrega a la lista el tiempo que tardó el programa en calcularlo
    y_iteractivo.append(tiempo)

    # Se imprimen los datos
    print(f'El numero {i} de Fibo es {fibb_ite} y tardó: {tiempo}[s]')
print("\n")
```

Conclusiones

El método de Fibonacci Iterativo es mucho más rápido que el método Fibonacci Recursivo, pues para calcular el número 40 el Iterativo le tomó menos de 1 segundo mientras que al Recursivo 33 segundos.

Esto es debido a que el método Iterativo es de complejidad lineal mientras que el Recursivo es de complejidad exponencial como se demuestra en las gráficas.