Tarea 1 Diseño y análisis de algoritmos

El algoritmo de ordenamiento burbuja se basa en el siguiente pseudocódigo:

```
para cada valor i de 1 a n
     para cada valor de i de 1 a n-1
          si el valor en la posición i es mayor al ubicado en la
          posición i+1
               intercambiar
```

Primeramente veremos la forma de crear un arreglo de números aleatorios de 1 a n, para ello importaremos la biblioteca ***random***

```
>>> import random
>>> random.randint(1,10)
6
>>> random.randint(1,10)
3
```

Con la sentencia ***import*** se importa la biblioteca y con la función ***randint(inicio, fin)*** creamos un número entero aleatorio entre inicio y fin.

De este modo podemos crear n números aleatorios:

```
import random

n = 10
numeros = []
for c in range(n):
 ale = random.randint(1, n)
 numeros.append(ale)
print(numeros)
```

y obtenemos algo parecido a esto:

```
[6, 4, 1, 5, 1, 4, 7, 3, 2, 8]
```

Aquí podemos notar que algunos números se pueden repetir, ya que no hemos puesto restricciones, esto no es un problema, pero si desean que no haya números repetidos se puede hacer lo siguiente:

```
# creamos el arreglo de aleatorios
numeros = []
while (len(numeros)<n):
 ale = random.randint(1,n)
 if not(ale in numeros):
 numeros.append(ale)
print(numeros)
```

y obtenemos algo como esto:

```
[7, 9, 1, 8, 2, 6, 10, 4, 5, 3]
```

Para evitar los números se repitan verificamos si ya está en el arreglo, si no está se agrega, de lo contrario no hacemos nada, como no sabemos cuántos números pueden salir repetidos utilizamos la sentencia ***while*** en lugar de ***for*** y nos detenemos hasta que tengamos n números en el arreglo.

Regresando al algoritmo de ordenamiento burbuja tenemos que la parte interna del algoritmo es:

```
para cada valor de i de 1 a n-1
          si el valor en la posición i es mayor al ubicado en la
          posición i+1
               intercambiar
```

Si tenemos 10 números, por ejemplo

```
[7, 9, 1, 8, 2, 6, 10, 4, 5, 3]
```

aplicamos el algoritmo tenemos:

```
[7, 9, 1, 8, 2, 6, 10, 4, 5, 3]
[7, 1, 9, 8, 2, 6, 10, 4, 5, 3]
[7, 1, 8, 9, 2, 6, 10, 4, 5, 3]
[7, 1, 8, 2, 9, 6, 10, 4, 5, 3]
[7, 1, 8, 2, 6, 9, 10, 4, 5, 3]
[7, 1, 8, 2, 6, 9, 4, 10, 5, 3]
[7, 1, 8, 2, 6, 9, 4, 5, 10, 3]
[7, 1, 8, 2, 6, 9, 4, 5, 3, 10]
[7, 1, 8, 2, 6, 9, 4, 5, 3, 10]
```

Se puede apreciar que los números grandes se van moviendo a la derecha, pero al final de la iteración, sólo podemos asegurar que un número se ha ordenado (el más grande), si queremos ordenarlos todos tenemos que repetirlo n veces, para lo cual agregamos un ciclo:

```
para cada valor i de 1 a n
     para cada valor de i de 1 a n-1
          si el valor en la posición i es mayor al ubicado en la
          posición i+1
               intercambiar
```

Si lo escribimos en Python como una función obtenemos:

```
import random

n = 10
def burbuja(lista):
 for i in range (len(lista)):
 for j in range(len(lista)-1):
 if (lista[j]>lista[j+1]):
```

```
 temp=lista[j]
 lista[j] = lista[j+1]
 lista[j+1] = temp
 return lista


# creamos el arreglo de aleatorios

numeros = []

while (len(numeros)<n):

 ale = random.randint(1,n)

 if not(ale in numeros):

 numeros.append(ale)

print(numeros)

print(burbuja(numeros))
```

y al ejecutarlo se obtiene:

```
[10, 9, 8, 1, 4, 5, 3, 2, 7, 6]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Se aprecian tanto el arreglo original como el ordenado, ¿pero que tan eficiente es?

El ciclo interno se ejecuta n-1 veces mientras que el ciclo externo se ejecuta n

veces:

```
def burbuja(lista):
    for i in range (len(lista)):
        for j in range(len(lista)-1):
            if (lista[j]>lista[j+1]):
                temp=lista[j]
                lista[j] = lista[j+1]
                lista[j+1] = temp
    return lista
```

es:

n-1
veces

n
veces

por lo que el costo de este algoritmo

Por lo que el costo depende de n^2, ese costo es muy elevado, para ejemplificarlo hay que tomar el tiempo de ejecución, para lo cual emplearemos la biblioteca time:

```
import time
inicio = time.time()
lista2 = burbuja(numeros)
fin = time.time()
print("el tiempo fue ", fin-inicio)
```

tomamos la hora del sistema antes de ordenar los números y después de haberlo hecho, restamos esos tiempos y obtenemos el tiempo total que ha tomado el ordenamiento, el programa completo queda así:

```
import random
import time

n = 10

def burbuja(lista):
 for i in range (len(lista)):
 for j in range(len(lista)-1):
 if (lista[j]>lista[j+1]):
 temp=lista[j]
 lista[j] = lista[j+1]
 lista[j+1] = temp
 return lista

# creamos el arreglo de aleatorios
numeros = []
while (len(numeros)<n):
 ale = random.randint(1,n)
 if not(ale in numeros):
 numeros.append(ale)
inicio = time.time()
lista2 = burbuja(numeros)
fin = time.time()

print("el tiempo fue ", fin-inicio)
```

si lo ejecutamos dice que el tiempo fue 0.0, ya que el tiempo que le lleva ordenar 10 números es muy pequeño, pero si en lugar de n = 10 lo cambiamos por n = 2500 aparece:

el tiempo fue 1.0452008247375488

En mi computadora se tarda un poquito más de un segundo para ordenar el arreglo, usted ajuste el valor de n para que sea lo más cercano a 1 seg en su computadora.

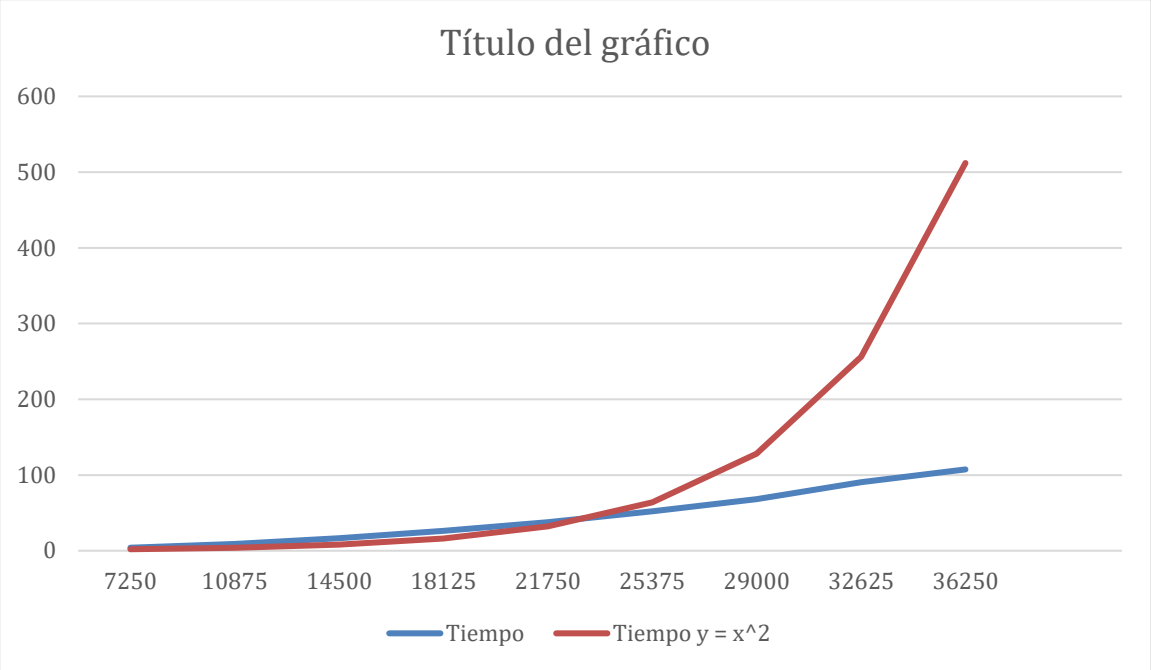
Ahora bien si en lugar de n=2500 pongo n=5000, que es el doble de tiempo, vemos lo que pasa con el tiempo:

el tiempo fue 4.2628068923950195

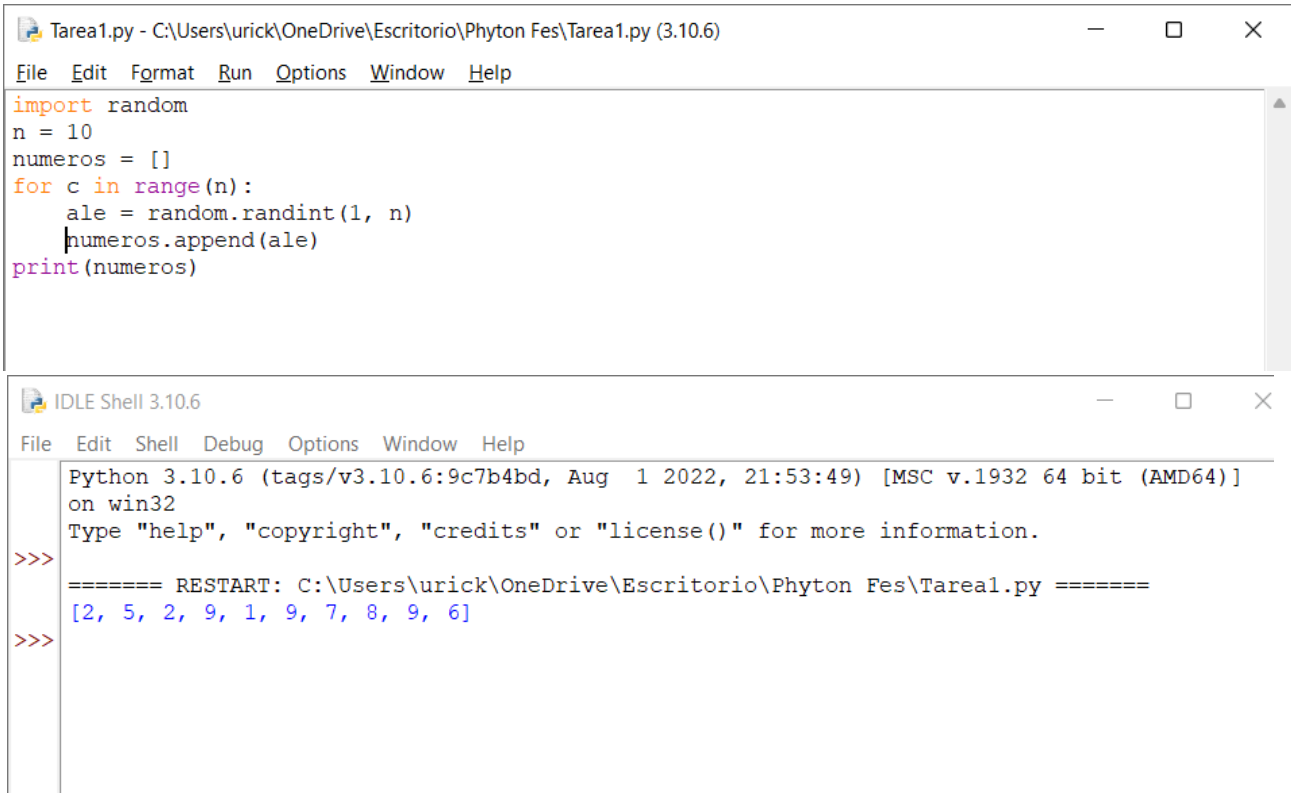
Cuando el valor es n = 7500, al ser el triple tarda aproximadamente 9 segundos, complete la siguiente tabla:

n	Tiempo
3625	1 seg
7250	4 seg
10875	9 seg
14500	16.5 seg
18125	26.4 seg
21750	37.7 seg
25375	52.02 seg
29000	68 seg
32625	90.5 seg
36250	107.4

Realice una gráfica con los valores obtenidos, compárela con la gráfica de y = x^2



Escriba una función ***esPar*** que al recibir un número devuelva verdadero si es par o falso si es impar, la tarea debe incluir la ejecución de todos los ejercicios, en word o pdf está bien.



Tarea1.py - C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py (3.10.6)

File Edit Format Run Options Window Help

```
import random
n = 10
numeros = []
for c in range(n):
    ale = random.randint(1, n)
    numeros.append(ale)
# creamos el arreglo de aleatorios
numeros = []
while (len(numeros)<n):
    ale = random.randint(1,n)
    if not(ale in numeros):
        numeros.append(ale)
print(numeros)
```

IDLE Shell 3.10.6

File Edit Shell Debug Options Window Help

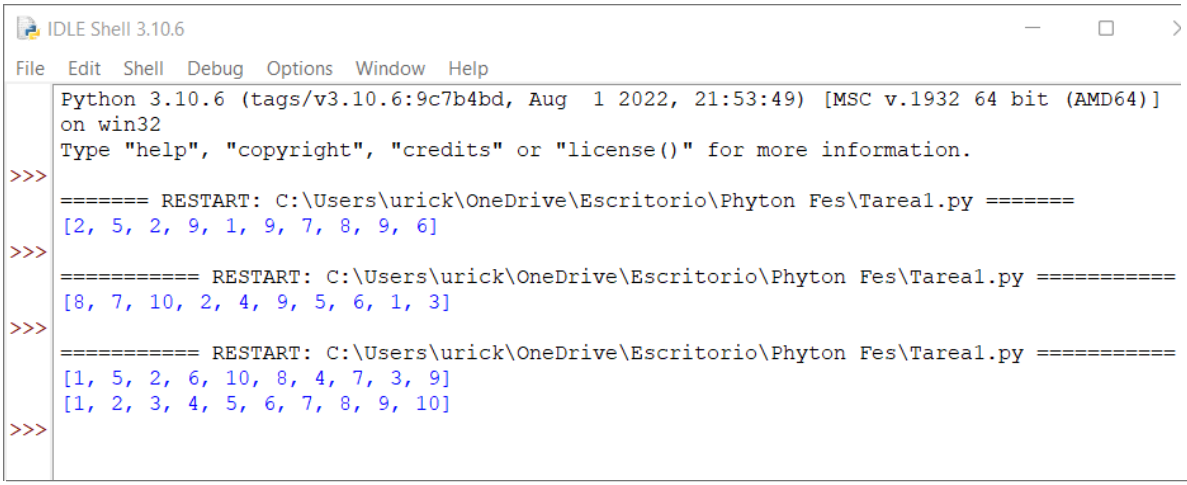
```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:53:49) [MSC v.1932 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py =======
[2, 5, 2, 9, 1, 9, 7, 8, 9, 6]
>>>
========== RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py ==========
[8, 7, 10, 2, 4, 9, 5, 6, 1, 3]
>>>
```

Tarea1.py - C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py (3.10.6)

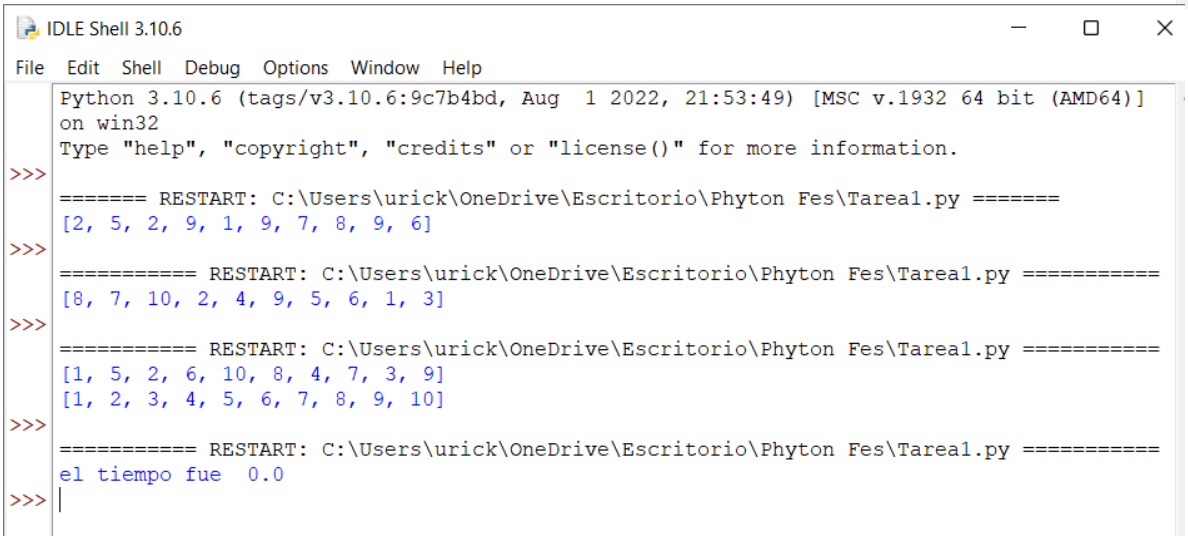
File Edit Format Run Options Window Help

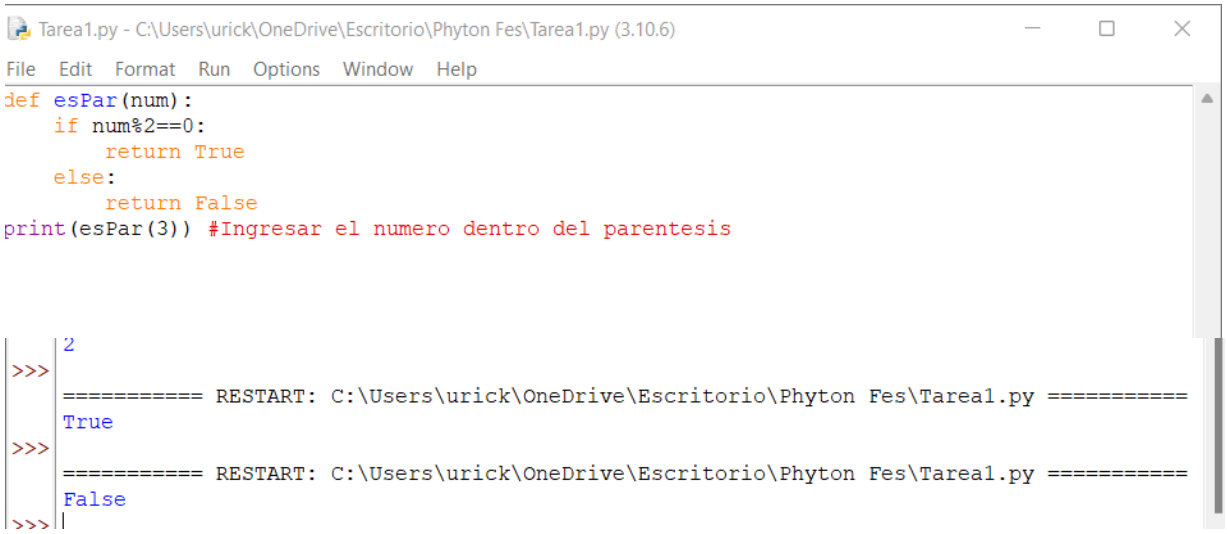
```
import random
n = 10
def burbuja(lista):

    for i in range (len(lista)):
        for j in range(len(lista)-1):
            if (lista[j]>lista[j+1]):
                temp=lista[j]
                lista[j] = lista[j+1]
                lista[j+1] = temp
    return lista
# creamos el arreglo de aleatorios
numeros = []
while (len(numeros)<n):
    ale = random.randint(1,n)
    if not(ale in numeros):
        numeros.append(ale)
print(numeros)
print(burbuja(numeros))
```


IDLE Shell 3.10.6
File Edit Shell Debug Options Window Help
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug 1 2022, 21:53:49) [MSC v.1932 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py =======
[2, 5, 2, 9, 1, 9, 7, 8, 9, 6]
>>>
========== RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py ==========
[8, 7, 10, 2, 4, 9, 5, 6, 1, 3]
>>>
========== RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py ==========
[1, 5, 2, 6, 10, 8, 4, 7, 3, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>>


Tarea1.py - C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py (3.10.6)
File Edit Format Run Options Window Help
import random
import time
n = 10
def burbuja(lista):
for i in range (len(lista)):
for j in range(len(lista)-1):
if (lista[j]>lista[j+1]):
temp=lista[j]
lista[j] = lista[j+1]
lista[j+1] = temp
return lista
# creamos el arreglo de aleatorios
numeros = []
while (len(numeros)<n):
ale = random.randint(1,n)
if not(ale in numeros):
numeros.append(ale)
inicio = time.time()
lista2 = burbuja(numeros)
fin = time.time()
print("el tiempo fue ", fin-inicio)


IDLE Shell 3.10.6
File Edit Shell Debug Options Window Help
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug 1 2022, 21:53:49) [MSC v.1932 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py =======
[2, 5, 2, 9, 1, 9, 7, 8, 9, 6]
>>>
========== RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py ==========
[8, 7, 10, 2, 4, 9, 5, 6, 1, 3]
>>>
========== RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py ==========
[1, 5, 2, 6, 10, 8, 4, 7, 3, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>>
========== RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py ==========
el tiempo fue 0.0
>>>


Tarea1.py - C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py (3.10.6)
File Edit Format Run Options Window Help
def esPar(num):
if num%2==0:
return True
else:
return False
print(esPar(3)) #Ingresar el numero dentro del parentesis
2
>>>
========== RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py ==========
True
>>>
========== RESTART: C:\Users\urick\OneDrive\Escritorio\Phyton Fes\Tarea1.py ==========
False
>>>