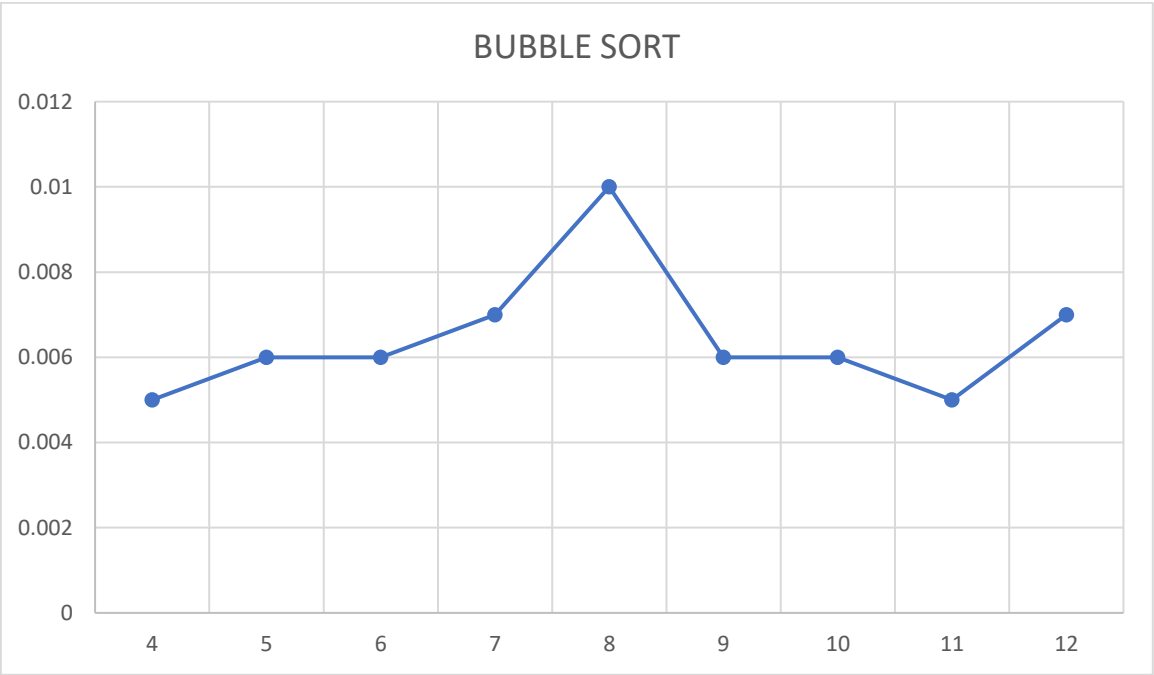
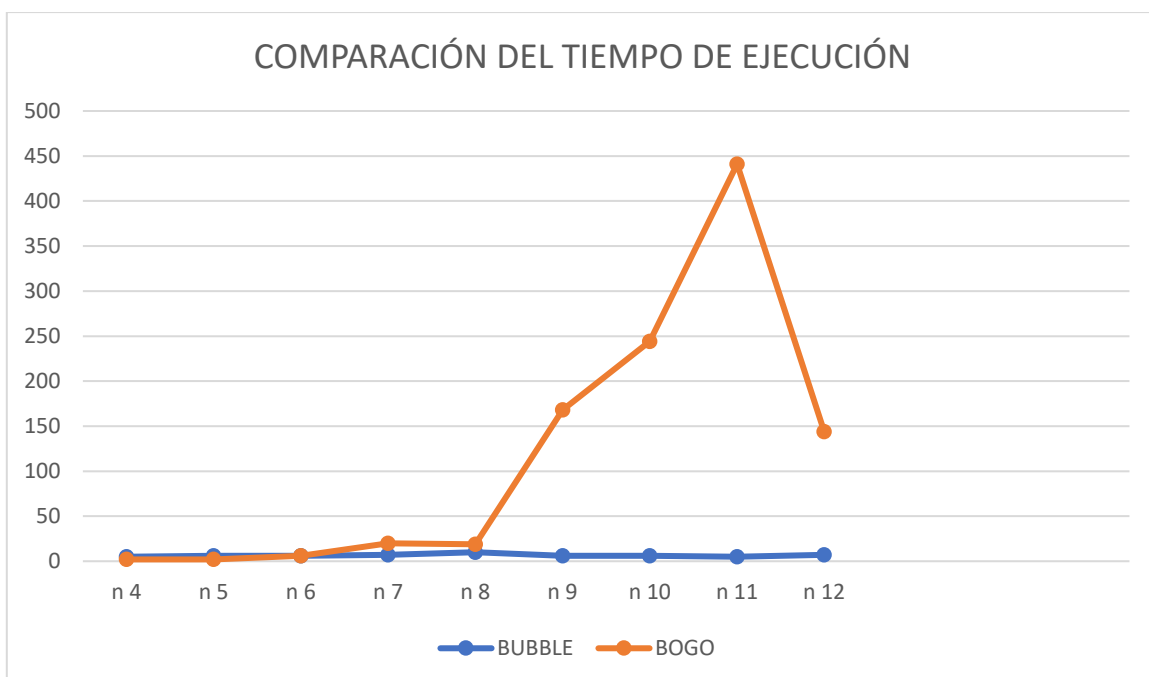
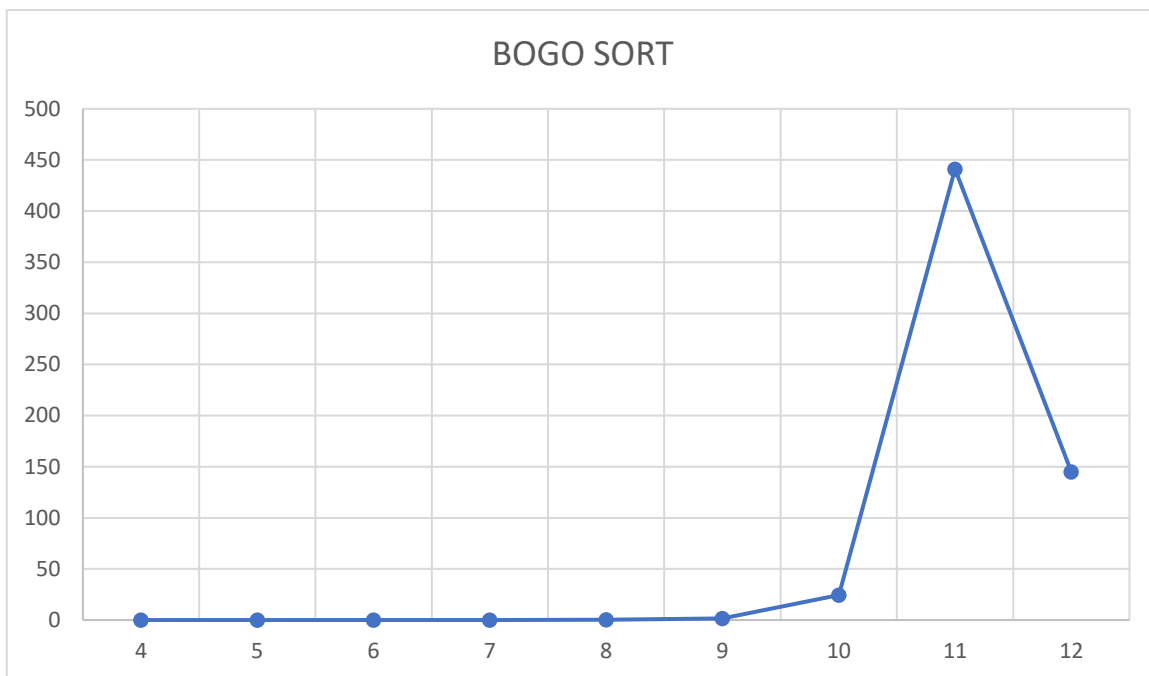


PARTE 1

n	Bubble Sort	Bogo Sort
4	0.005	0.002
5	0.006	0.002
6	0.006	0.006
7	0.007	0.020
8	0.010	0.19
9	0.006	1.68
10	0.006	24.48
11	0.005	441.11
12	0.007	144.99





Parte 2

n	Bubble Sort
10	0.005
100	0.005
1000	0.16
10000	17.127
100000	---
1000000	---
10000000	---
100000000	---

Esta prueba es la mas pesada de todas ya que aunque el Bubble Sort es bueno para el acomodo de datos, observamos que mientras mas aumenta la cantidad de números lo hace del mismo modo el tiempo que lleva acomodarlos

Para este punto los 100000 el calculo fue imposible para mi computadora ya que el programa solo parecía no hacer nada, se noto un aumento de velocidad y ruido en el ventilador interno, asi como un aumento en el porcentaje de procesamiento que requiere el CMD



Procesos						
Rendimiento		Historial de aplicaciones	Inicio	Usuarios	Detalles	Servicios
Nombre	Estado		91%	62%	41%	0%
			CPU	Memoria	Disco	Red
>  Procesador de comandos de Wi...			24.8%	10.2 MB	0 MB/s	0 Mbps
>  Procesador de comandos de Wi...			24.1%	9.5 MB	0 MB/s	0 Mbps

Ilustración 1 ADMINISTRADOR DE TAREAS DURANTE EJECUCIÓN

```
C:\Users\johan\Desktop\ATOMPYTHON>BUBBLE.py
1661921920.47
-
```

Ilustración 2 PROGRAMA BUBBLE.py CORRIENDO ATRAVEZ DE CMD EN ESTADO DETENIDO

BUBBLE.py — C:\Users\johan\Desktop\ATOMPYTHON — Atom

File Edit View Selection Find Packages Help

Project

- ATOMPYTHON
 - BOGO.py
 - BUBBLE.py
 - SUMA1
 - SUMA1.py
 - SUMA2.py

SUMA1.py

```
1 import time
2 from random import random
3 from random import randint
4 #from future import print_function
5
6 def burbuja( data):
7     n = len(data)
8     for iteracion in range( n ):
9         for pivote in range( n - iteracion - 1):
10             if data[pivote] > data[pivote +1]:
11                 tmp = data[pivote]
12                 data[pivote]=data[pivote+1]
13                 data[pivote+1]=tmp
14         print(data)
15
16 ini = time.time()
17 print(time.time())
18
19
20 original = []
21
22 for _ in range (100000):
23     original.append(randint(1,100000))
24
25 burbuja(original)
26
27 print(time.time())
28 fin = time.time()
29 final= (fin - ini)
30 print("milisegundos")
```

BOGO.py

Ilustración 2 CODIGO CON EJECUCION DE $n = 100000$