UNIVERSIDAD NACIONAL DE SAN AGUSTÍN
FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS
ANÁLISIS Y DISEÑO DE ALGORITMOS - B

GUIA DE LABORATORIO

DOCENTE: ALEX FLOREZ.

ALUMNA: QUISPE QUISPE GABRIELA MALENA
GRUPO B- LABORATORIO

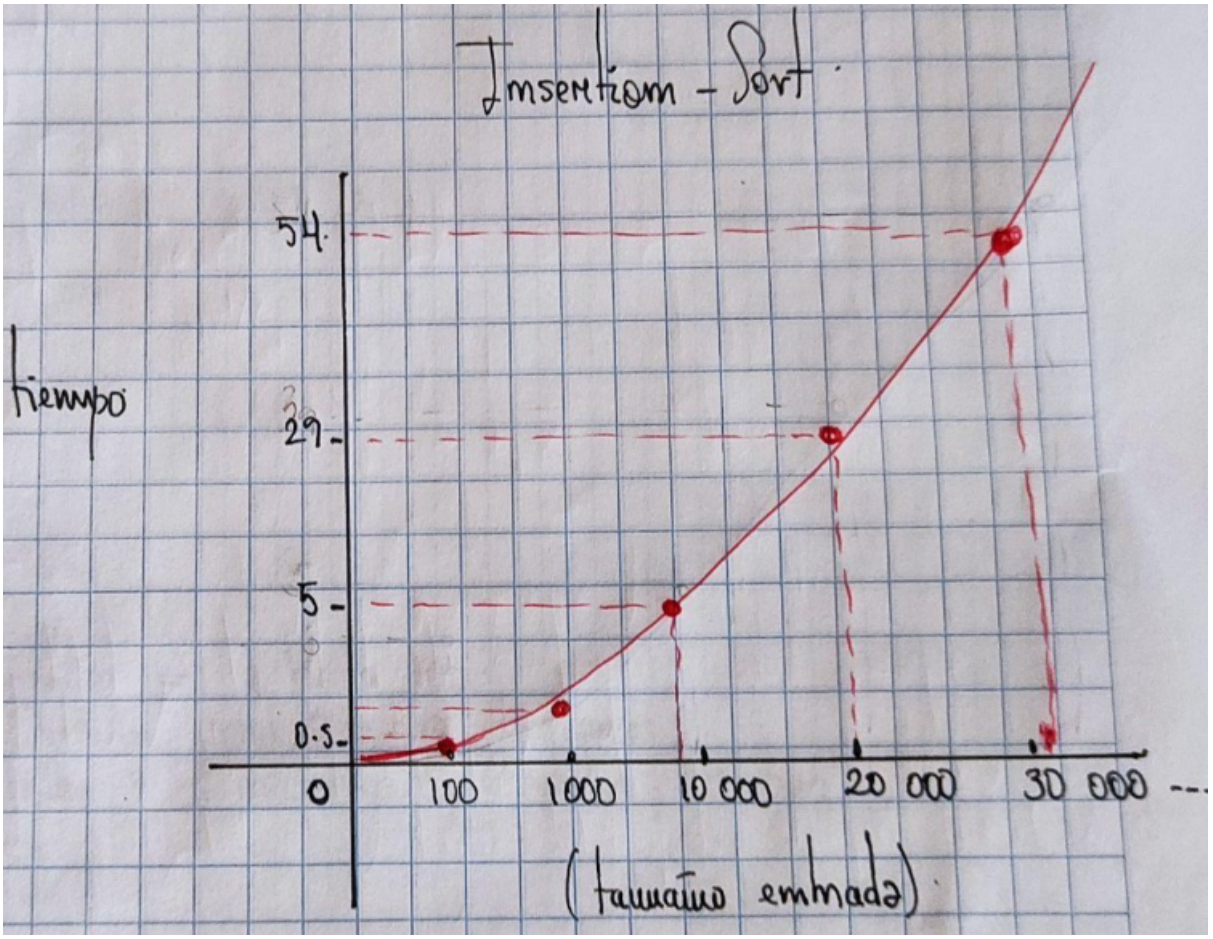
UNSA
UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

A Continuación graficamos cada una de las gráficas de los algoritmos trabajados, de Insertion Sort, Búsqueda Secuencial, Búsqueda Binaria, así como su tiempo de ejecución:

A. GRÁFICA USANDO TIEMPO EJECUCIÓN - INSERTION SORT

En este caso indicamos el rango va aumentando de 100, 1000, etc al lado nos muestra el tiempo de ejecución de cada rango y en la parte inferior la gráfica.

RANGO	TIEMPO DE EJECUCIÓN
(0,10000)	5.8158204555511475
(0,20000)	29.745809316635132
(0,30000)	54.360745668411255
(0,40000)	90.56535601615906
(0,50000)	146.32818508148193
(0,100)	0.0005316734313964844
(0,1000)	0.06238842010498047




```
<terminated> InsertionSort [Java Application] C:\Users\Gabriela Quispe\.p2\pool\plugins\org.eclips
El arreglo ingresado es:[0, 6, 2, 4, 1, 9, 12]
El arreglo ordenado Insertion es: :[0, 1, 2, 4, 6, 9, 12]

SU TIEMPO DE EJECUCION ES : 2714000 nanosegundos.
```

DemoBusquedaBinaria.java | BusquedaBinaria.java

```java
//En este ejercicio ingresamos nuestro
//Ordena nuestro arreglo de manera asc

public class InsertionSort {
    static long tiempoInicio;
    static long tiempoFinal;


    public static void main(String[] args) {
        tiempoInicio= System.nanoTime();
        int[] arreglo = {0,6,2,4,1,9,12};  //declaramos nuestro array
        System.out.println("El arreglo ingresado es:" + Arrays.toString(arreglo));
        for(int i=1; i<arreglo.length-1; i++)   {  // ordenamos por insercion
            int temporal = arreglo[i];
            int j= i-1;
            while(j>=0 && temporal <= arreglo[j])   {
                arreglo[j+1] = arreglo[j];
                j = j-1;
            }
            arreglo[j+1] = temporal;
        }
        System.out.println("El arreglo ordenado Insertion es: :" + Arrays.toString(arreglo));
        tiempoFinal = System.nanoTime();
        //Resta del tiempo final menos el tiempo de inicio
        System.out.println("\nSU TIEMPO DE EJECUCION ES : " + (tiempoFinal-tiempoInicio) + " nanosegundos.\n")
```

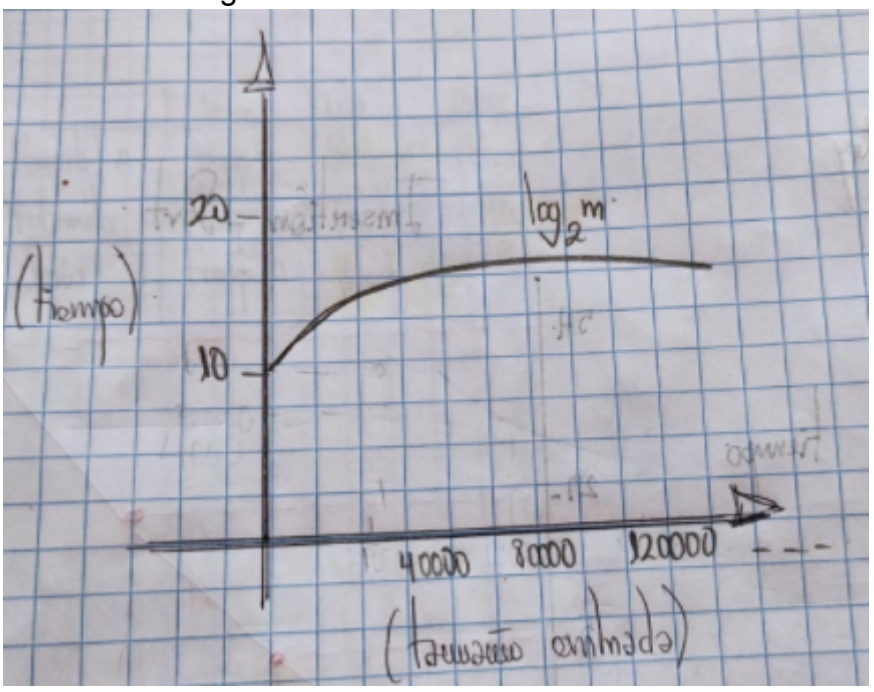
También desarrolle el Insertion Sort con su tiempo de ejecución en Python, en el anterior utilice java ahora veremos como funciona en Python:

```
main.py
'''

#Ejercicio 4- hallar el tiempo de ejecucion
#del insertion sort realizado en python
# utilizo numeros random van de 10,100,1000, 20 000...

from random import randint
from time import time

arreglo= []
for i in range(0,1000):
    arreglo.append (randint(0,100000))

#print(arreglo)

tiempo_inicial = time()
for i in range(1,len(arreglo)) :
    valor = arreglo[i]
    j=i-1
    while j>=0 and valor < arreglo[j]:
        arreglo[j+1] = arreglo[j]
        j= j-1
    arreglo[j+1] = valor

tiempo_final = time()
#print(arreglo)
print("El tiempo de ejecucion es : "+str(tiempo_final-tiempo_inicial))
```

```
El tiempo de ejecucion es : 0.11567902565002441
```

B. GRÁFICA USANDO TIEMPO EJECUCIÓN - BUSQUEDA BINARIA

Desarrolle en java el ejercicio de búsqueda binaria, usando también tiempo de ejecución en nanosegundos.



```java
public class BusquedaBinaria {
    static long tiempoInicio;
    static long tiempoFinal;

    public static int[] LlenarArreglo() {
        tiempoInicio= System.nanoTime();
        int arreglo[]= new int[10]; // indicamos nueswtra canti
        for (int i=0; i<arreglo.length; i++) {
            arreglo[i]= (int)
        }
        return arreglo;
    }
```

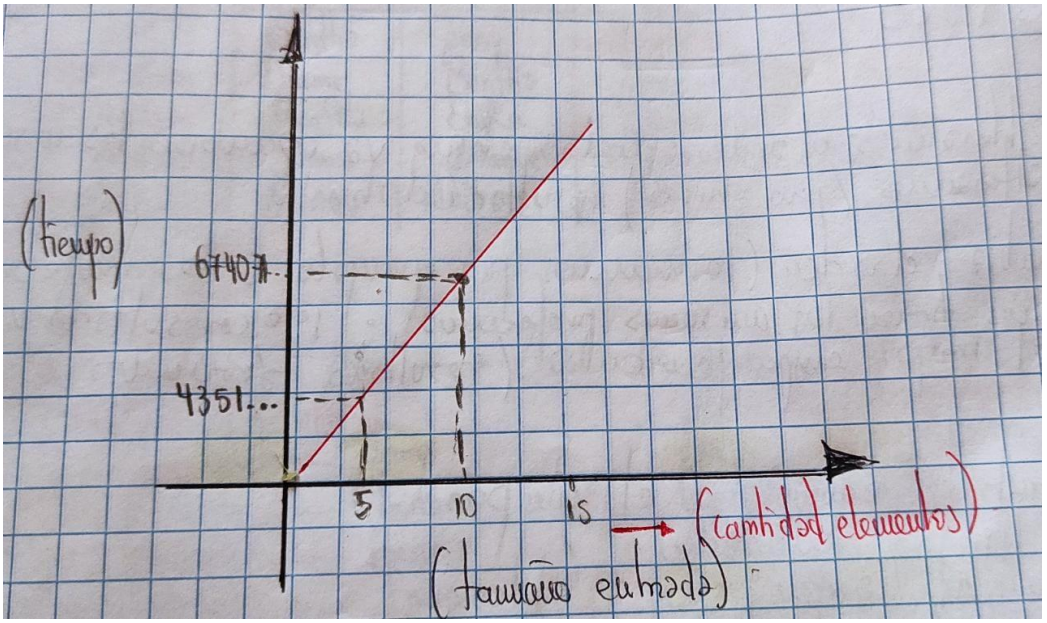
Console

<terminated> DemoBusquedaBinaria (1) [Java Application] C:\Users\Gabri

```
SU TIEMPO DE EJECUCION ES : 3700 nanosegund

posicion0 numero 7
posicion1 numero 664
posicion2 numero 1357
posicion3 numero 2549
posicion4 numero 3482
posicion5 numero 4088
posicion6 numero 4650
posicion7 numero 5291
posicion8 numero 8025
posicion9 numero 8568
numero que desea buscar?
8025
Su numero es: 8025 y existe en el arreglo:
```

C. GRÁFICA USANDO TIEMPO EJECUCIÓN - BUSQUEDA SECUENCIAL

En este caso el tiempo de ejecución los desarrolle en java,



noBusquedaBinaria.java | BusquedaBinaria.java | BusquedaSecuencial.java | InsertionSort.java

```java
public class BusquedaSecuencial {
    static long tiempoInicio;
    static long tiempoFinal;


    public static void main(String[] args) {
        tiempoInicio= System.nanoTime();
        int [] num = {2,4,1,5,9,12,33}; //nuestr
        Scanner scan = new Scanner(System.in);
        boolean presente= false; // si el numero

        System.out.println("Ingrese número:"); // indicamos num
        int encontrarNum = scan.nextInt();
        for(int x = 0; x < num.length; x++){
            if(num[x]== encontrarNum){
                System.out.println("True, posicion "+(x+1));
                break;
            }
            if(x == num.length-1){
                presente = true;
            }
        }
        if(presente==true){
            System.out.println("false");
        }

        tiempoFinal = System.nanoTime();
        //Resta del tiempo final menos el tiempo de inicio
        System.out.println("\nSU TIEMPO DE EJECUCION ES : " + (tiempoFinal-tiempoInicio) + " nanosegundos.
    }
```

Console

<terminated> BusquedaSecuencial [Java Application] C:\Users\Gabriela Quispe\.p2\pool\

```
Ingrese número:
5
True, posicion 4

SU TIEMPO DE EJECUCION ES : 2944521100 nanosegundos.
```