

## **Documentación Práctica 3**

### **Estructura de Datos Lineales**

#### **Algoritmos iterativos de búsqueda y ordenamiento.**

#### **Ordenamiento por Inserción y Búsqueda Binaria.**

##### **Introducción.**

El ordenamiento por inserción es un algoritmo de ordenamiento con complejidad de tiempo de  $O(n)$  en su mejor caso, siendo "n" la cantidad de elementos en la lista, en el peor de los casos la complejidad de tiempo del algoritmo es de  $O(n^2)$ . El ordenamiento por inserción ordena los elementos en una lista un elemento a la vez, conforme se agregan elementos se compara el nuevo elemento con los elementos ordenados de la lista y lo inserta hasta que termine de realizar comparaciones. Este algoritmo de ordenamiento no es muy eficiente si se requiere ordenar listas muy grandes.

La búsqueda binaria o búsqueda logarítmica es un algoritmo de búsqueda que encuentra la posición de un elemento en una lista. Este algoritmo parte la lista a la mitad y compara el elemento que se encuentra en la mitad con el elemento, "llave", que se quiere encontrar, si este es igual se ha encontrado el elemento, sino y el elemento de la mitad es mayor que la llave, la mitad de la lista se convierte en el final de la lista y se repite el proceso, si el elemento de la mitad es menor que la llave entonces la mitad de la lista se convierte en el inicio de la lista y se repite el proceso. La complejidad de este algoritmo es de  $O(\log n)$  siendo "n" la cantidad de elementos en la lista.

##### **Descripción del Problema.**

Con el uso de un método que genere números aleatorios se crea una lista doblemente ligada de tamaño N, cada vez que se genere un número este se debe insertar usando ordenamiento por inserción hasta que los elementos se acaben, al finalizar se mostrara en pantalla el tiempo que tardo en realizar el ordenamiento.

Con la lista ya ordenada se procederá a buscar una llave usando búsqueda binaria y mostrar en pantalla si el elemento fue encontrado o no y cuánto tiempo tardo en realizar la búsqueda.

##### **Desarrollo.**

A continuación, se mostrarán los pseudocódigos de los métodos que se implementarán para resolver el problema.

##### **Pseudocódigo estructura nodo:**

- NODO:  
entero iDato  
Nodo pnSig.  
Nodo pnAnt.

### **Pseudocódigo Inserción Ascendente:**

- Entrada: E  
Salida: I

```
Si (pnLista <> 0) entonces
    pnLista <- E;
Otro
    pnAux2 <- pnLista.pnAnt
    pnAux <- pnLista.
    Mientras (pnAux <> 0 y E.iDato > pnAux.iDato) Hacer
        pnAux2 <- pnAux
        pnAux <- pnAux.pnSig
    Fin Mientras
    Si (pnAux2 <> 0) entonces
        pnAux2.pnSig <- E
    Otro
        pnLista <- E
    Fin Si
    E.pnAnt <- pnAux2
    Si (pnAux <> 0) entonces
        pnAux.pnAnt <- E
    Fin Si
    E.pnSig <- pnAux
Fin Si
pnContador <- pnLista
Regresa pnLista;
```

### **Pseudocódigo Recorre:**

- pnAux  
pnAux <- pnContador  
Si (pnContador = 0) entonces  
 pnContador <- pnLista  
Otro  
 pnContador = pnContador.pnSig  
regresa pnAux

### **Pseudocódigo Mitad de la Lista:**

- Entrada: pnInicio, pnFinal  
Salida: pnMitad

```
Nodo pnMitad, pnFin
Si (pnInicio == 0) entonces
    Regresa 0
Fin Si
pnMitad <- pnInicio
pnFin <- pnInicio.pnSig
```

```

Mientras (pnFin <> pnFinal) Hacer
    pnFin <- pnFin.pnSig
    Si (pnFin <> pnFinal) entonces
        pnMitad <- pnMitad.pnSig
        pnFin <- pnFin.pnSig
    Fin Mientras
    Regresa pnMitad

```

### **Pseudocódigo Búsqueda Binaria:**

- Entrada: pnLista, llave  
Salida: Nulo/pnMitad

```

pnInicio <- pnLista
pnFinal <- 0
Repetir
    pnMitad = mitadLista(pnInicio, pnFinal)
    Si (pnMitad = 0) entonces
        Regresa 0
    Fin Si
    Si (pnMitad.iDato = llave) entonces
        Regresa pnMitad
    Sino Si (pnMitad.iDato < llave) entonces
        pnInicio <- pnMitad.pnSig
    Otro
        pnFinal <- pnMitad
Mientras (pnFinal = 0 o pnFinal <> pnInicio)
    Regresa 0

```

**Número aleatorio:** Para generar un número aleatorio se usa una función propia de C++ llamada rand().

### **Resultados:**

Con un ciclo “para” se crea uno por uno los nodos con su respectivo número aleatorio y se hace uso del método inserciónAscendente después de que se crea un nodo, el ciclo se repite de 0 hasta N, siendo N el número de nodos que se desean crear. En esta práctica se crearon 100,000 nodos.

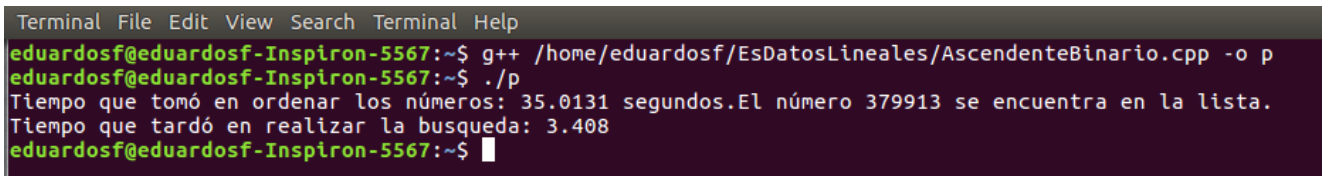
Para conocer el tiempo que le tomo al método realizar el ordenamiento hacemos uso de la función clock() de la librería <ctime> de c++, e inicializamos dos variables de tipo entero una que guarde el tiempo inicial y otra el tiempo final, hay que considerar que esta función usa el reloj de la computadora para tomar el tiempo. Ya con estas dos variables inicializadas colocamos la variable de tiempo inicial antes del ciclo “para” y la de tiempo final después del ciclo y colocamos un “cout” para que nos muestre en segundos el tiempo que le tomo al método completar el ordenamiento.

Realizado el ordenamiento se inicializa una variable de tipo entero que tendrá como nombre “llave” y le asignamos algún valor, este valor se usará como parámetro junto con la lista

ordenada en el método “busquedaBinaria” para conocer si la llave proporcionada se encuentra o no en la lista. Si se encuentra se mostrará en pantalla que la llave se encuentra en la lista y el tiempo que tardo en encontrarlo, para conocer el tiempo se hace uso de las variables que guardan el tiempo que previamente usamos y se colocan antes del método de busca binaria y otro después de este; Si la llave no fue encontrada se mostrara en pantalla que la llave no se encuentra en la lista y el tiempo que tardo en realizar la búsqueda.

A continuación, se presenta una imagen de la terminal de Linux con el resultado del problema que muestra:

- El tiempo que tardó en realizar el ordenamiento por inserción.
- Si encontró o no encontró la llave en la lista y cuánto tiempo tardó en realizar la búsqueda.



```
Terminal File Edit View Search Terminal Help
eduardosf@eduardosf-Inspiron-5567:~$ g++ /home/eduardosf/EsDatosLineales/AscendenteBinario.cpp -o p
eduardosf@eduardosf-Inspiron-5567:~$ ./p
Tiempo que tomó en ordenar los números: 35.0131 segundos.El número 379913 se encuentra en la lista.
Tiempo que tardó en realizar la búsqueda: 3.408
eduardosf@eduardosf-Inspiron-5567:~$
```

### Manual de Usuario:

- Para compilar el programa es necesario escribir lo siguiente en la terminal de LINUX:
  - `g++ /home/nombreUsuario/ubicaciónArchivo/nombreArchivo.cpp -o p`
  - Siendo “nombreArchivo” el nombre que lleva el programa, en este caso lleva por nombre: AscendenteBinario
- Para ejecutar el programa se introduce lo siguiente:
  - `./p`
  - La última letra de la línea de compilación indica la letra que va después del “./”.