Documentación Práctica 4.
Estructura de Datos Lineales
Algoritmos iterativos de búsqueda y ordenamiento.
Ordenamiento por Inserción y Ordenamiento Burbuja.

Introducción.

El ordenamiento por inserción se investigó en la práctica anterior, por lo tanto, en esta se escribirá del ordenamiento burbuja.

El ordenamiento burbuja es uno de los algoritmos de ordenamiento más sencillos que funciona mediante el cambio repetido de dos elementos adyacentes que se encuentran en un orden incorrecto. Este algoritmo es de complejidad $O(n^2)$ lo que lo hace ineficiente con grandes volúmenes de información, ya que siempre checa el orden de los elementos desde el principio de la lista, aunque estos ya hayan pasado por el ordenamiento. Un problema más de este algoritmo de ordenamiento es que el tiempo que le toma ordenar todos los elementos depende de la manera en la que los elementos están acomodados inicialmente, por ejemplo:

- Se tiene una lista acomodada de la siguiente manera: {2, 3, 4, 5, 1}, al algoritmo le tomará $O(n^2)$ iteraciones para ordenar la lista.
    - 2 > 3, correcto, 3 > 4 correcto, 4 > 5 correcto, 5 > 1, cambio.
    - 2 > 3, correcto, 3 > 4 correcto, 4 > 1, cambio.
    - 2 > 3, correcto, 3 > 1 cambio.
    - 2 > 1, cambio.
    - {1, 2, 3, 4, 5}. Lista Ordenada.
- Se tiene una lista acomodada de la siguiente manera: {5, 1, 2, 3, 4}, al algoritmo le tomará O(n) iteraciones para ordenar la lista.
    - 5 > 1, cambio.
    - 5 > 2, cambio.
    - 5 > 3, cambio.
    - 5 > 4, cambio.
    - 1 > 2, correcto, 2 > 1, correcto, 3 > 4 correcto, 4 > 5, correcto.
    - {1, 2, 3, 4, 5}. LIsta Ordenada.

Como es evidente el tiempo depende del orden inicial de la lista por lo que mientras más elementos tenga la lista y menos ordenada esté, más tardado el ordenamiento.

Descripción del problema.

Con el uso de un método que genera números aleatorios se crearán listas simples doblemente ligadas y listas circulares doblemente ligadas de tamaño N. En las listas que usen método de ordenación por inserción se creará el nodo e inmediatamente se ordenará; en el caso de las listas que usen el método de ordenamiento burbuja primero se creará la lista completa y posteriormente se ordenará. También se usará el método de búsqueda binaria que se usó en la práctica pasada. Las listas deberán de ser ordenadas de manera ascendente y descendente por ambos métodos.

Desarrollo.

A continuación, se mostrarán los pseudocódigos de los métodos que se implementarán para resolver el problema, en este desarrollo ya no se incluirán los pseudocódigos de los métodos de ordenamiento por inserción y búsqueda binaria para la lista doblemente ligada, así como el método que encuentra la mitad de la lista. Tampoco se mostrarán los pseudocódigos de la estructura de un nodo, los métodos de insertar nodo, recorrer lista, y la del número aleatorio.

Pseudocódigo Inserción Descendente Lista Doblemente ligada:

- Entrada: E
  Salida: I

```
Si (I<> 0) entonces
  I<- E
Otro
  pnAux2 <- I.Ant
  pnAux <- I
  Mientras (pnAux <> 0 y E.Dato < pnAux.Dato) entonces
    pnAux2 <- pnAux
    pnAux <- pnAux.Sig
  Fin Mientras
  Si(pnAux2 <> 0) entonces
    pnAux2.Sig <- E
  Otro
    I <- E
  Fin Si
  E.Ant <- pnAux2
  Si(pnAux <> 0)
    pnAux.Ant <- E
  Fin Si
  E.Sig <- pnAux
Fin Si
pnContador <- I
Regresa I
```

Pseudocódigo Ordenamiento Burbuja Ascendente/Descendente:

- Entrada: I
  Salida: pnAux

```
Boleano bandera <- true
pnAux = I
Mientras(bandera) Hacer
 bandera <- falso
 pnAux2 <- pnAux
 Si(pnAux.Dato > (pnAux.Sig).Dato y pnAux) entonces
   temp <- pnAux.Sig
 Otro
   temp <- pnAux
 Fin Si
 Mientras(pnAux2) Hacer
   Si(comparaAscendente(pnAux, pnAux.Sig) <> 1) entonces // Esta linea puede contener el método comparaAscendente o el método comparaDescendente.
     pnAux2 <- pnAux2.Sig
   Otro
     bandera <- true
 Fin Mientras
Fin Mientras
```

Regresa pnAux

Pseudocódigo Compara Ascendente:

- Entrada: E, E2
  Salida: entero 1 o 0

```
Si(E == 0 ó E2 == 0) entonces
  Regresa 0
Si(E.Dato > E2.Dato) entonces
  Si(I == E) entonces
    I <- E2
  Fin Si
  Si(E.Ant <> 0) entonces
    (E.Ant).Sig <- E2
  Fin Si
  E.Sig <- E2.Sig
  Si(E2.Sig <> 0) entonces
    (E2.Sig).Ant <- E
  Fin Si
  E2.Ant <- E.Ant
  E2.Sig <- E
  E.Ant <- E2
  Regresa 1
Otro
  Regresa 0
Fin Si
```

Pseudocódigo Compara Descendente:

- Entrada: E, E2
  Salida: entero 1 o 0

```
Si(E == 0 ó E2 == 0) entonces
  Regresa 0
Si(E.Dato < E2.Dato) entonces
  Si(I == E) entonces
    I <- E2
  Fin Si
  Si(E.Ant <> 0) entonces
    (E.Ant).Sig <- E2
  Fin Si
  E.Sig <- E2.Sig
  Si(E2.Sig <> 0) entonces
    (E2.Sig).Ant <- E
  Fin Si
  E2.Ant <- E.Ant
  E2.Sig <- E
  E.Ant <- E2
  Regresa 1
Otro
  Regresa 0
Fin Si
```

Pseudocódigo Inserción Ascendente Lista Circular Doblemente Enlazada.

- Entrada: E
  Salida: I

```
Si(! I) entonces
  E.Sig <- E
  E.Ant <- E
  pnLista <- E
Otro
  Si(I.Dato > E.Dato) entonces
    Si(I.Ant == I) entonces
      I.Sig <- E
      E.Sig <- I
      I.Ant <- E
      I <- E
    Otro
      (I.Ant).Sig <- E
      E.Ant <- I.Ant
      E.Sig <- I
      I.Ant <- E
      I <- E
    Fin Si
  Sino Si(I.Dato < E.Dato y E.Dato > (I.Ant).Dato)
    (I.Ant).Sig <- E
    E.Sig <- I
    E.Ant <- I.Ant
    I.Ant <- E
  Otro
    pnAux2 <- I.Ant
    pnAux <- I
    Mientras(E.Dato > pnAux.Dato y pnAux.Sig <> I) entonces
      pnAux2 <- pnAux
      pnAux <- pnAux.Sig
    Fin Mientras
    pnAux2.Sig <- E
    E.Sig <- pnAux
    pnAux.Ant <- E
    E.Ant <- pnAux2
  Fin Si
Fin Si
pnContador <- I
Regresa I
```

Pseudocódigo Inserción Descendente Lista Circular Doblemente Enlazada.

- Entrada: E
  Salida: I

```
Si(! I) entonces
  E.Sig <- E
  E.Ant <- E
  pnLista <- E
Otro
  Si(I.Dato <= E.Dato) entonces
    Si(I.Ant == I) entonces
      I.Sig <- E
      E.Sig <- I
      I.Ant <- E
      I <- E
    Otro
      (I.Ant).Sig <- E
      E.Ant <- I.Ant
      E.Sig <- I
      I.Ant <- E
      I <- E
    Fin Si
  Sino Si(I.Dato > E.Dato y E.Dato < (I.Ant).Dato)
    (I.Ant).Sig <- E
    E.Sig <- I
    E.Ant <- I.Ant
    I.Ant <- E
  Otro
    pnAux2 <- I.Ant
    pnAux <- I
    Mientras(E.Dato < pnAux.Dato y pnAux.Sig <> I) entonces
      pnAux2 <- pnAux
      pnAux <- pnAux.Sig
    Fin Mientras
    pnAux2.Sig <- E
    E.Sig <- pnAux
    pnAux.Ant <- E
    E.Ant <- pnAux2
  Fin Si
Fin Si
pnContador <- I
Regresa I
```

Pseudocódigo Compara Ascendente Lista Circular Doblemente Enlazada.

- Entrada: E, E2
  Salida: entero 1 ó 2

```
Si(E y E2) entonces
  Si(E.Dato > E2.Dato) entonces
    Si(E == I)
      I = E2
    Fin Si
    E.Sig <- E2.Sig
    (E2.Sig).Ant <- E
    E2.Ant <- E.Ant
    (E.Ant).Sig <- E2
    E2.Sig <- E
    E.Ant <- E2
    Regresa 1
  Otro
    Regresa 0
Fin Si
```

Pseudocódigo Compara Descendente Lista Circular Doblemente Enlazada.

- Entrada: E, E2
  Salida: entero 1 ó 2

```
Si(E y E2) entonces
  Si(E.Dato < E2.Dato) entonces
    Si(E == I)
      I = E2
    Fin Si
    E.Sig <- E2.Sig
    (E2.Sig).Ant <- E
    E2.Ant <- E.Ant
    (E.Ant).Sig <- E2
    E2.Sig <- E
    E.Ant <- E2
    Regresa 1
  Otro
    Regresa 0
Fin Si
```

Pseudocódigo Búsqueda Binaria Lista Circular: Ascendente y Descendente.

- Entrada: pnLista, llave
  Salida: Nulo/pnMitad

```
pnIncio <- pnLista
pnFinal <- pnInicio.Ant
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 //Para cambiar la búsqueda a Descendente se cambia la condición a : Si (pnMitad.Dato > llave).
    pnIncio <- pnMitad.pnSig
  Otro
    pnFinal <- pnMitad
Mientras (pnFinal = pnInicio.Ant o pnFinal <> pnInicio)
Regresa 0
```

Pseudocódigo Ordenamiento Burbuja Lista Circular Doblemente Enlazada: Ascendente y Descendente.

- Entrada: I
  Salida: I

```
bandera <- verdadero
pnAux <- I
Mientras(bandera) entonces
  bandera <- falso
  pnAux2 <- pnAux
  Mientras (pnAux2.Sig <> I) entonces
    Si (comparaAscendente/Descendente (pnAux2, pnAux2.Sig) <> 1) entonces
      pnAux2 <- pnAux2.Sig
    Otro
      Bandera <- verdadero
  Fin Mientras
  PnAux <- I
Fin Mientras.
```

//El pseudocódigo para el ordenamiento burbuja es el mismo tanto para el ordenamiento ascendente como para el descendente, lo único que cambia es el método que se usa para comparar.

Resultados:

Con un ciclo “para” se crea un por uno los nodos con su respectivo número aleatorio. Esto se hará para cada tipo de lista la única diferencia radica en la manera en la que se ordenaran los nodos, si los nodos se ordenan usando ordenamiento por inserción estos se estarán ordenando conforme se crean, mientras que si se decide ordenarlos usando ordenamiento burbuja primero

se creará la lista completa y posteriormente se ordenará. Al usuario se le dará la opción de elegir entre ordenar una lista de manera ascendente o de manera descendente, con la ayuda de una función switch se podrá ordenar la lista de la manera que eligió el usuario.

También se le dará la opción al usuario de buscar en la lista un número el que sea que desee encontrar. Si el número aparece en la lista aparecerá un mensaje en pantalla indicando que el número se encuentra en la lista o en caso contrario indicando que no se encuentra.

El proceso de pedir al usuario que elija la manera en la que desea ordenar las listas y pedirle un número que desea encontrar se hará cuatro veces dos para la lista doblemente enlazada y dos para la lista circular doblemente enlazada, y el proceso de buscar un número en la lista se hará cuatro veces, uno por cada vez que la lista sea ordenada.

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: insercionBurbuja

- Para ejecutar el programa se introduce lo siguiente:
    - ./p
    - La última letra de la línea de compilación indica la letra que va después del “ ./ ”.
- Cuando el programa le pida al usuario presionar 1 o 2 para elegir entre ordenar de manera ascendente o descendente, el usuario deberá teclear una sola vez el número 1 o 2 y presionar la tecla Enter, si el usuario llegase a presionar otra tecla y presiona Enter el programa termina.
- El programa cuando requiera una llave para la búsqueda binaria trabaja estrictamente con números enteros, en caso de que el usuario teclee otra cosa diferente a un entero el programa marcará un error pues no incluye excepciones a casos ajenos al entero.