Documentación Práctica 5.
Estructura de Datos Lineales
Algoritmos iterativos de búsqueda y ordenamiento.
Quicksort.

Introducción.

Quicksort es un algoritmo de ordenamiento del tipo “Divide y Conquista”. El cual toma un elemento como pivote y divide la fila alrededor del pivote. Existen diferentes versiones en Quicksort que nos permiten elegir un pivote:

- Siempre elegir el primer elemento como el pivote.
- Siempre elegir el último elemento como el pivote.
- Elegir un elemento al azar como el pivote.
- Elegir la mitad de la lista como pivote.

El elemento clave en el algoritmo de ordenamiento Quicksort es el pivote y como a partir de este los elementos de la fila se acomodan. En esta versión de “Quicksort” se trabajará con un pivote y a partir de este recorrer toda la fila en busca de los elementos menores que el pivote e insertarlos antes de este, y así, dejando a todos los mayores al pivote después del pivote, y repetir el proceso hasta que ambas sub-filas estén ordenadas. En la siguiente práctica veremos otro método para ordenar los números con ayuda del pivote y dos auxiliares, el inicio de la fila y el fin de la fila.

La complejidad de tiempo de este algoritmo de ordenamiento es de $O(n^2)$ en su peor caso y de O (log n) en su mejor caso, lo cual quiere decir que es uno de los algoritmos de ordenamiento más rápidos que existen.

Descripción del problema.

Con el uso de un método que genera números aleatorios se creará una lista (fila) doblemente ligada de tamaño N, y con ayuda del algoritmo de ordenamiento “Quicksort” ordenaremos la lista en forma ascendente. Al finalizar el ordenamiento es necesario que se muestre en pantalla el tiempo que le tomo al algoritmo ordenar los elementos de la fila.

Desarrollo.

A continuación, se mostrarán los pseudocódigos de los métodos que se implementarán para resolver el problema, en este desarrollo no se incluirán los métodos de la Pila, pues está se vio previamente en la práctica 2 y su implementación es básicamente la misma. Tampoco se incluirán los pseudocódigos de la fila doble para los métodos de inserción e insertar, pero si se mostrarán los métodos de insertar antes del pivote y extraer elementos mayores al pivote. De igual manera no se incluirá el pseudocódigo del método que genera números aleatorios.

En este desarrollo se incluyó un tipo de nodo diferente al que usualmente usamos, este nodo recibe como parámetros dos apuntadores de tipo nodo y un apuntador a nodo siguiente, pero este de tipo nodoP, esto como referencia al pivote, pues este guarda los apuntadores del inicio de la sub-fila(s) y el fin de la sub-fila(s).

Pseudocódigo insertar antes del pivote.

- Entrada: pivote, E
  Salida: I;

  Si (! pivote.Ant) entonces
    I <- E;
  Otro
    (pivote.Ant)->pnSig <- E
  Fin Si
  E.Sig <- pivote
  E.Ant <- pivote.Ant
  Pivote.Ant <- E

Pseudocódigo extraer elementos mayores al pivote.

- Entrada: E
  Salida: E

  Si (! E.Ant) entonces
    I <- E.Sig
  Otro
    (E.Ant).Sig <- E.Sig
  Fin Si
  Si(E.Sig) entonces
    (E.Sig).Ant <- E.Ant
  Otro
    Final <- E.Ant
  Fin Si
  E.Sig <- 0
  E.Ant <- 0
  Regresa E

Pseudocódigo crear nodoP.

- Entrada: nodo inicio, nodo fin.
  Salida: E

  E <- nuevo nodoP
  E.Inicio <- inicio
  E.Final <- fin
  E.Sig <- 0
  Regresa E

Pseudocódigo establecer inicio de la fila.

- Entrada: I
  Salida: I
  Esto.I <- I

Pseudocódigo obtener inicio fila.

- Entrada: I
  Salida: E
  Retorna I

Pseudocódigo establecer final de la fila.

- Entrada: I
  Salida: I
  Final <- I
  Mientras (Final.Sig) entonces
    Final <- Final.Sig

Pseudocódigo obtener final de la fila.

- Entrada: I
  Salida: E
  Retorna E;

Pseudocódigo ordenamiento "Quicksort".

- Entrada: I
  Salida: I
  Funcion Vacia quickSort(I)
    Llama FILA Fila //Llamada a clase FILA se instancia Fila
    Llama PILA Pila //Llamada a clase PILA se instancia Pila

    Inicio <- I
    Si(Inicio) entonces
      Fila.InicioFila(Inicio) //Llamada a función establecer inicio fila.
      Fila.FinFila(); //Llamada a función establecer final fila.
      Fin <- Fila.ObtenerFinal //Llamada a función obtener final fila.
      Puntero <- Fila.CrearNodoP(Inicio, Fin) //Llamada a función crear NodoP.
      Pila.PUSH(Puntero) //Llamada a función Push de pila.
      Mientras (Pila <> 0) Hacer
        Puntero <- Pila.POP //Llamada a función POP de pila.
        fila <- puntero.Inicio
        Fin <- puntero.Final
        Pivote <- fila
        Bandera <- verdadero
        Mientras (Bandera) Hacer
          Aux <- Aux.Sig
          Si (Aux = Fin.Sig) entonces
            Bandera <- Falso
          Otro
            Si (Aux.Dato < pivote.Dato) entonces
              Si (Aux = Fin) entonces
                Fin <- Fin.Ant
              Fin Si
              Aux <- Aux.Ant
              Ext <- Fila.Extraer(AuxSig) //Llamada a función extraer.
              Fila.InsertarAntesPivote(Pivote, Ext) //Llamada a función insertar.
              Si (Pivote = fila) entonces
                Fila <- Ext
              Fin Si
          Fin Si
        Fin Mientras

        Si (fila <> Fin) entonces

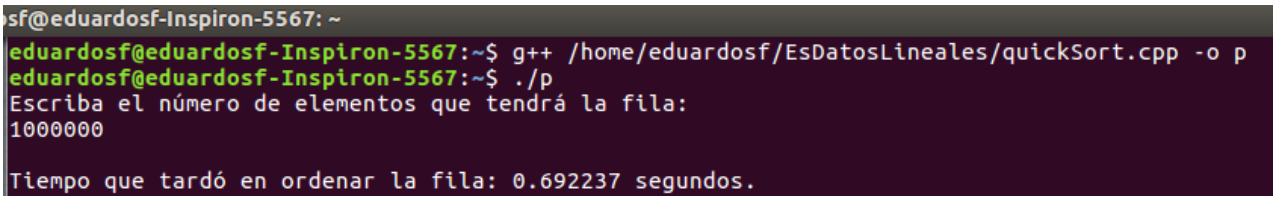
```
        Si (fila.Sig <> Fin) entonces
          Si (Pivote <> fila) entonces
            Puntero <- Fila.CrearNodoP(fila, pivote.Ant) //Llamada a función crear NodoP.
            Pila.PUSH(Puntero) //Llamada a función PUSH Pila.
          Fin Si
          Si (Pivote <> Fin) entonces
            Puntero <- Fila.CrearNodoP(Pivote.Sig, Fin) //Llamada a función crear NodoP.
            Pila.PUSH(Puntero) //Llamada a función PUSH Pila.
          Fin Si
        Fin Si
      Fin Mientras
      I <- Fila.InicioFila //Llamada a función obtiene el inicio de la fila.
    FinProcedimiento
```

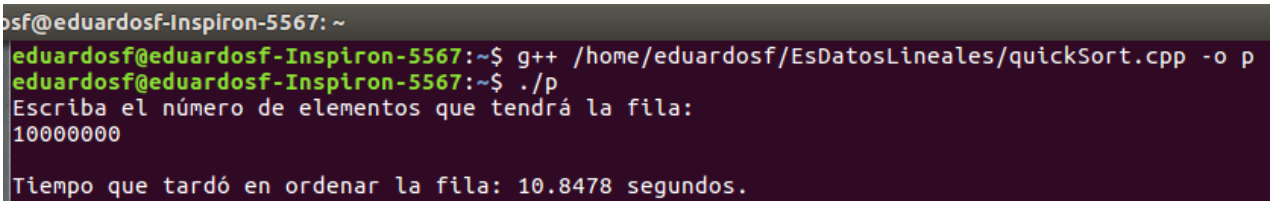
Resultados.

Con un ciclo “para” se crean uno por uno los nodos con su respectivo número aleatorio y se hace uso del método que inserta nodos en la fila, esto hasta que terminé de crear la cantidad de nodos que el usuario indicó, ya que al ejecutar el programa aparecerá en consola la indicación al usuario de proporcionar la cantidad de nodos que tendrá la fila. Ya con todos los nodos creados e insertados en la fila se hará uso del método quickSort, el cual ordenará los nodos de manera ascendente. Finalizada la ejecución del método aparecerá en consola el tiempo en segundos que le tomó al método en realizar el ordenamiento.

En un entorno real el usuario no podría modificar el código para que muestre la lista, pero en este caso se hará para darle certeza a quien revisa el código que el método si ordena los elementos de la fila correctamente. Para esto se deberá des-comentar las líneas 294, 295, 301 y 302 del código, se recomienda que no se des-comente si la cantidad de nodos excede mil elementos pues no será posible mostrarlos todos, pero si se requiere puede hacerse.

A continuación, se mostrarán una serie de imágenes que nos demuestran la efectividad de este algoritmo al ordenar una gran cantidad de elementos:





```
sf@eduardosf-Inspiron-5567: ~
eduardosf@eduardosf-Inspiron-5567:~$ g++ /home/eduardosf/EsDatosLineales/quickSort.cpp -o p
eduardosf@eduardosf-Inspiron-5567:~$ ./p
Escriba el número de elementos que tendrá la fila:
100000000

Tiempo que tardó en ordenar la fila: 178.406 segundos.
```

Como se puede observar el tiempo que le tomo al algoritmo ordenar grandes volúmenes de elementos es muy poco si lo comparamos con otros métodos de ordenamiento, por ejemplo, al algoritmo de ordenamiento por inserción le toma 33 segundos ordenar 100,000 elementos mientras que al algoritmo de ordenamiento Quicksort le toma menos de 1 segundo ordenar un millón de elementos.

Claramente se puede ver la diferencia en la efectividad que tiene este algoritmo y a pesar de que el número de nodos que tiene que ordenar es demasiado grande al algoritmo le toma menos de 3 minutos en ordenar 100 millones de elementos.

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: quickSort
  - Si la ubicación del archivo se encuentrá en descargas la línea para compilar se vería de la siguiente manera:
    - g++ /home/nombreUsuario/Downloads/quickSort.cpp -o p
- Para ejecutar el programa se introduce lo siguiente:
  - ./p
  - La última letra de la línea de compilación indica la letra que va después del “ ./ ”.