Documentación Práctica 9
Estructura de Datos Lineales
Recursividad
Merge Sort/Ordenamiento por Mezcla

Introducción.

“Merge Sort” es un algoritmo del tipo “Divide y Conquista”. El cual funciona dividiendo la lista inicial y la divide hasta llegar al caso base, el caso base se obtiene de dividir la lista en sub-listas y hasta obtener una sub-lista con uno o dos elementos, es aquí cuando obtenemos nuestro caso base.

Ya con el caso base se procede a ordenarlo, haciendo una simple comparación e intercambiar los elementos si el elemento “a” es menor o mayor que “b”. Esto se hace hasta obtener dos sub-listas y se procede a ordenar las sub-listas mediante el método “Merge Sort”.

Este algoritmo es de complejidad “O (n log n)” tanto en su mejor y peor caso, siendo n el número de elementos de la lista.

Descripción del problema.

Se generará una fila de nodos, cada nodo contendrá una llave, esta llave se generará de un arreglo de números aleatorios, por ende, al finalizar la creación de la fila sus llaves estarán desordenadas; y con el uso del algoritmo “Merge Sort” se tendrá que ordenar la fila.

Desarrollo.

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

Los métodos de insertar nodo, crear nodo, conocer el número de nodos, recorrer la fila y saber la mitad de la lista se implementaron en el desarrollo de la práctica, pero no se mostrarán en esta sección, ya que fueron usados en prácticas anteriores, esto también aplica para la estructura NODO.

Pseudocódigo Inserta Nodo Mezcla.

- Entrada: E
  Salida: I

```
Funcion Vacío insertarMezcla (nodo E)
   Aux <- I
   Aux2 <- Final
   Si (I = 0)
      I <- E
      Final <- E
   Otro
      Si (I.Llave <= Final.Llave) entonces
         E.Ant <- Final
         Aux2.Sig <- E
         Final <- E
      Otro
         E.Sig <- I
         Aux.Ant <- E
         I <- E
      Fin Si
   Fin Si
Fin Función
```

Pseudocódigo Extrae Nodo Mezcla.

- Entrada: E, E2
  Salida: E

```
Función Nodo ExtraeMezcla(nodo E, nodo E2)

   E2 <- E2.Sig
   Si(E2) entonces
      E2.Ant <- 0
   Fin Si
   E.Sig <- 0
   E.Ant <- 0
   Retorna E
Fin Función
```

Pseudocódigo ordena mitad.

- Entrada: E, E2
  Salida: Fila

```
Función Vacío ordenaMitad(nodo E, nodo E2)
   EAnt <- E
   E2Ant <- E2
   Mientras (EAnt y E2Ant) Hacer
      Si (EAnt.Llave > E2Ant.Llave) entonces
         E2Ant <- Llamada a función ExtraeMezcla(E2Ant, E2)
         Llamada a función insertarMezcla (E2Ant)
      Otro
         EAnt <- Llamada a función ExtraeMezcla(EAnt, E)
         Llamada a función insertarMezcla(EAnt)
      Fin Si
      EAnt <- E
      E2Ant <- E2
   Fin Mientras
   Mientras (E2Ant) Hacer
      E2Ant <- Llamada a función ExtraeMezcla(E2Ant, E2)
      Llamada a función insertarMezcla (E2Ant)
      E2Ant <- E2
   Fin Mientras
   Mientras (EAnt) Hacer
      EAnt <- Llamada a función ExtraeMezcla(EAnt, E)
      Llamada a función insertarMezcla (EAnt)
      EAnt <- E
   Fin Mientras
Fin Función
```

Pseudocódigo Intercambia Nodo.

- Entrada: E
  Salida: I

```
Función Vacío Intercambio(nodo E)
   Si (E.Sig || E.Ant) entonces
      Si(E.Llave > (E.Sig).Llave) entonces
         E.Ant <- E.Sig
         E.Sig <- 0
         (E.Ant).Sig <- E
         (E.Ant).Ant <- 0
         I = E.Ant
      Fin Si
   Fin Si
Fin Función
```

Pseudocódigo Merge Sort.

- Entrada: inicio, N, E
  Salida: I

```
Función Vacío mergeSort(entero inicio, entero N, E)
   nodo a, A
   a <- E
   entero mitad
   mitad <- (inicio + N) / 2

   Si (inicio <> N e inicio + 1 <> N) entonces
      E <- Llamada a función MitadFila(E, 0)
      A <- E.Sig
      A.Ant <- 0
      Llamada a función mergeSort (inicio, mitad, a)
      Llamada a función mergeSort (mitad + 1, N, A)
      Mientras (a.Ant <> 0) Hacer
         a <- a.Ant
      Fin Mientras
      Mientras (A.Ant <> 0) Hacer
         A <- A.Ant
      Fin Mientras
   Otro
      Llamada a función Intercambia (E)
   Fin Si
Fin Función
```

Pseudocódigo solución Merge Sort.

- Entrada: I, i, Final, E
  Salida: I

```
Inicio Función solMergeSort()
   Nodo E
   Entero i <- 1
   Entero Final <- Llamada a Función NumeroNodos()
   E <- I
   Si (0 <= Final) entonces
      Llamada a función mergeSort(i, Final, E)
   Fin Si
Fin Función
```

Resultados.

Se crea una fila de nodos con su respectiva llave, cada llave es generada a partir de un arreglo con números ordenados de 0 a N y con ayuda de dos índices auxiliares se mezclan los números y se insertan de nuevo en el arreglo.

Con la lista creada se invoca el método solMergeSort() en el main para que se inicie el ordenamiento. Al finalizar la ejecución la lista estará ordenada.

A continuación, se muestra un ejemplo de la implementación del método:

Fila Desordenada:

18 28 45 2 10 21 1 3 51 17 11 9 15 7 12

Fila Ordenada:

1 2 3 7 9 10 11 12 15 17 18 21 28 45 51

Manual de Usuario.

- ara 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: mergeSort.cpp
- Para ejecutar el programa se introduce lo siguiente:
  - ./p
  - La última letra de la línea de compilación indica la letra que va después del “ ./ ”.