Proyecto #1: Ordenamiento Externo

Universidad Nacional Autónoma de México
Facultad de Ingeniería

Alumnos

Aguilera Roa, Mauricio Arturo
Núñez Trejo, Emilio

Materia

Estructura de Datos y Algoritmos II

24 de marzo del 2019

0.1 Marco teórico

El objetivo de este proyecto es la implementación de 3 algoritmos de ordenamiento externo, junto con el desarrollo de habilidades en el manejo de archivos y la programación que esto conlleva.
Este tipo de algoritmos son de gran funcionalidad en el mundo moderno, donde las problematicas en cualquier campo, presentan más de diez elementos como variables a ordenar; por ello es importante aprender la implementación de dichos algoritmos.
El ordenamiento de datos en la computación requiere una método a seguir, y un estudio previo de las necesidades a cubrir. Al existir tantos métodos de ordenamiento, se debe seleccionar con un análisis detallado el algoritmo que se va a implementar; una de las razones principales, es el costo de implementación. El costo de implementación juega el rol principal al momento de escoger un algoritmo de ordenamiento. Cómo ejemplo, no podemos aplicar el mismo algoritmo de ordenamiento a una empresa que ordena en promedio 50 datos por día, con una qué almacena más de 1,000 datos nuevos por día.

0.2 Antecedentes

Ordenamiento en ciencias de la computación se refiere al acomodo, ya sea de forma descendente o ascendente de datos, listas o archivos. Se considera qué existen dos tipos de ordenamiento primordiales:

1. **Algoritmos de ordenamiento interno**
   Hacen uso de la memoria principal. Todos los objetos, listas o archivos que se ordenan permanecen en la memoria RAM.
   Ejemplos:

   - Bubble Sort
   - Quick Sort
   - Radix Sort

2. **Algoritmos de ordenamiento externo**
   El ordenamiento se realiza mediante la transferencia de bloques con información hacia la memoria principal, una vez ordenado el bloque, este se regresa a memoria secundaria.
   Ejemplos:

   - Polifase
   - Mezcla Equilibrada
   - Distribución

Para el desarrollo de este proyecto, unicamente se hará enfásis en los 3 algoritmos de ordenamiento externos, mencionados previamente (Polifase, Mezcla Equilibrada y Distribución).

Polifase

Este algoritmo hace uso de archivos auxiliares para almacenar el resultado parcial. La estrategia que utiliza, es mezclar cuantas veces sean necesarias, hasta vaciar el archivo original. A la par de la ejecución, se mezcla tanto el archivo de entrada, como el último de salida, para intercambiar elementos. Construyendo al final, el archivo ordenado.

Mezcla Equilibrada

Podría ser considerado como una optimización del algoritmo *Mezcla Directa*. A diferencia de Mezcla Directa, las particiones en Mezcla Equilibrada se toman en secuencias ordenadas de máxima longitud. La mezcla se realiza entre las secuencias en forma alternada sobre dos archivos diferentes.

Distribución

A diferencia de otros algoritmos de ordenamiento externos, éste se basa en el algoritmo de ordenamiento interno *Radix Sort*. El Radix Sort utilizado en el almacenamiento interno, hace uso de colas para cada elemento de la cadena, y extrapola lo anterior. Mientras qué el Radix Sort de almacenamiento externo utiliza archivos para cada símbolo analizado en la cadena.

0.3 Análisis

Polifase

- Herramientas
  Esta sección de código es la encargada de importar las bibliotecas que permiten el uso de funciones especializadas para el programa. Además de ello, contiene el package denominados OrdenamientosExternos, qué engloba toda el programa.

- Clase Principal
  En esta parte se incluye todo el código funcional para esta clase de Java, se podría denominar como la clase principal de Polifase.

- Ordenar
  Polifase hace uso de Merge Sort para ordenar los elementos del archivo, esta función esta destinada solamente al proceso de ordenamiento.

- Polifase
  Se construye una instancia de la clase Polifase, inicializando el atributo manejadorArchivo con filePath, creando dos archivos auxiliares e inicializando manejadores con las direcciones de éstos.

- Resetear Lectores
  Con esta función se regresa al lector de cada manejador de archivo, la posición 0 del archivo, es parte fundamental del ordenamiento, ya que sin resetear los lectores, sobreescribiríamos el archivo.

- Limpiar Archivos
  Al igual que Resetear Lectores, Limpiar Archivos es parte fundamental del ordenamiento, se encarga de vaciar los archivos creados temporalmente, traspasar la información.

- Borrar Archivos
  Durante el proceso de ordenamiento se crean archivos temporales, mediante Borrar Archivos se eliminan, evitando generar archivos basura.

- Mostrar Dirección de Archivos
  Esta función nos permite saber donde se alojó el archivo que ordenamos, sobreescribe el archivo, pero la dirección a pesar de ser la misma qué en un inicio, la devuelve.

Mezcla Equilibrada

- Herramientas
  Esta sección de código es la encargada de importar las bibliotecas que permiten el uso de funciones especializadas para el programa. Además de ello, contiene el package denominados OrdenamientosExternos, qué engloba toda el programa.

- Clase Principal
  En esta parte se incluye todo el código funcional para esta clase de Java, se podría denominar como la clase principal de Mezcla Equilibrada, por ello contiene los dos archivos temporales, en los que almacena los elementos; simulando F0 y F1.

- Mezcla Equilibrada
  Construye una instancia de la clase Mezcla Equilibrada. Se inicializa el atributo manejadorArchivo con filePath, a la vez se crean los dos archivos que simulan F0 y F1, y para concluir su funcionamiento inicializa los manejadores de los archivos con sus respectivas direcciones.

- Ordenar
  Esta es la clase destinada al ordenamiento de los elementos, con la dirección que el FilePath le otorgó, de esta manera se asegura la dirección del archivo a ordenar.

- Crear Bloques Ordenados
  Se le puede considerar la parte fundamental de Mezcla Equlibrada, para que funcione como tal. Como su nombre lo dice, se encarga de segmentar los elementos en bloques ordenados y los almacena en archivos auxiliares.

- Resetear Lectores
  Con esta función se regresa al lector de cada manejador de archivo, la posición 0 del archivo, es parte fundamental del ordenamiento, ya que sin resetear los lectores, sobreescribiríamos el archivo.

- Limpiar Archivos Temporales
  Cómo la función Limpiar Archivos en Polifase, Limpiar Archivos Temporales, se encarga de vaciar los archivos F0 y F1 que se crearon para ordenar los elementos.

- Borrar Archivos Temporales
  Se encarga de eliminar los archivos temporales, los cuales previamente fueron vaciados con la función Limpiar Archivos Temporales.

- Mostrar Dirección de Archivos
  Esta función nos permite saber donde se alojó el archivo que ordenamos, sobreescribe el archivo, pero la dirección a pesar de ser la misma qué en un inicio, la devuelve.

Distribución

- Herramientas
  Esta sección de código es la encargada de importar las bibliotecas que permiten el uso de funciones especializadas para el programa. Además de ello, contiene el package denominados OrdenamientosExternos, qué engloba toda el programa.

- Clase Principal
  En esta parte se incluye todo el código funcional para esta clase de Java, se podría denominar como la clase principal de Distribución. A diferencia de las clases principales de los otros algoritmos, está contiene 10 archivos temporales, en los cuales almacena los elementos de acuerdo a su terminación númerica: 0, 1, 2, 3, 4, 5, 6, 7, 8 o 9.

- Radix Sort Externo
  Hace uso de filePath para leer la dirección del archivo a ordenar, además de crear los 10 archivos temporales, mencionados previamente.

- Ordenar
  Esta es la clase destinada al ordenamiento de los elementos, con la dirección que el FilePath le otorgó, de esta manera se asegura la dirección del archivo a ordenar.

- Borrar Archivos Temporales
  Se encarga de borrar los archivos temporales, los cuales traspasaron su información mediante el manejador de archivos.

- Mostrar Dirección de Archivos
  Esta función nos permite saber donde se alojó el archivo que ordenamos, sobreescribe el archivo, pero la dirección a pesar de ser la misma qué en un inicio, la devuelve.