InformeProyecto

Universidad Nacional Autonoma de México

25 de marzo de 2019

Estructura de Datos y Algoritmos II

Grupo: 5

Semestre: 2019-2

Profesor: M.C. Tista García Edgar

Integrantes:

- Calzada Martínez Jonathan Omar
- Jiménez Hernández Juan Carlos
- Garcia Lazcano Carlos David

Introducción

Los algoritmos de ordenamiento externo se usan cuando hay que manejar grandes cantidades de información. Estos algoritmos entran en función cuando la información a ordenar no cabe en la memoria principal del sistema; entonces se recurre al uso de una memoria mas grande, aunque esto conlleva a un proceso de ordenamiento mas lento.

En estructura de Datos y Algoritmos es muy importante como administrar los elemntos que se nos dan,datos, en este caso es el ordenamineto de estos mediante álgoritmos que usan archivos externos como apoyo para realizar estas labores.

Un buen código consiste de lógica y una correcta traducción al lenguaje que se este usando , en nuestro caso usaremos Java; el cual, ademas de ser totalmente gratuito, cuenta con una guía que nos permite verificar el funcionamineto de cada una de sus librerias.

En su mayoría los ordenamientos externos realiza una primera pasada sobre los archivos a ordenar, dividiéndolo en bloques de tamaño manipulable en la memoria principal. En segundo lugar ordena cada bloque con algún método interno, y por último mezcla los bloques ordenados realizando varias pasadas sobre el archivo.

Dado que el costo principal de los ordenamientos externos está dado por el tiempo de acceso al dispositivo, los algoritmos buscan reducir el número de pasadas sobre el archivo. En estos casos se tiene que definir el número máximo de valores que se pueden llevar a memoria principal, número total de llaves a ordenar y la cantidad de archivos utilizados en los métodos.

Desarrollo

Se nos encargo realizar tres algoritmos de ordenamiento, que utilizan archivos de tipo texto para ordenar de manera eficiente los datos que se encuentran en un archivo en específico.

Sí bien no existe el archivo, desde un inicio, se logro implementar que no solo se cree un documento, tambien se inicializan carpetas y los archivos auxiliares con el nombre de este y subfijos que los diferencía de los otros.

BufferedReader: este funciona para leer los archivos de un texto
BufferedWritter: funciona para escribir en un archivo
PrintWriter Imprime en pantalla lo que se ha escrito en el programa
FileWritter: escribe en un fichero, el cual pertenece a un archivo el cual nosotros hemos creado dentro de una carpeta

Distribución

Se dice que este método nació de la idea de Herman Hollerith en 1890 al crear la maquina tabuladora, en la cual se empleaban tarjetas perforadas para realizar el censo de ese año en Estados Unidos. Al final, despues de unas horas, la maquina entregaba todo un grupo de hojas listas para ser procesadas en un computador. En el censo de 1880 se tomaron 10 años para procesar toda la información, pero con las tarjetas perforadas, en la maquina que incluía un "card sorter" se tomaron cerca de 6 semanas. La idea original de Hollerith era ordenar empezando por el digito más significativo.
De esta manera se usa el algoritmo, pero en nuestro caso se utiliza Strings, y al ir seprando las cadenas, según sea el caracter a ordenar se encolara al archivo correspondiente, para despues leer el archivo en el que se almaceno y volver a escribirlo en el archivo original, así hasta terminar con el ciclo, cuando se realice la ultima iteración se escribirá en el archivo original, y este será el documento finalmente ordenado.

Polifase

En este álgoritmo se usan m cantidad de archivos, esta parte se la dejaremos que la utilice el usuario, y dependiendo el numero de m que introduzaca se iran distribuyendo los elementos del documento a ordenar en estos, una vez distribuidos lo que procede es, con un algoritmo de ordenamiento interno, se irán acomodando por bloques desde un elemento y se va concatenado con otros, hasta formar el arreglo ordenado, sé consideran varias cosas para lograr esto, en nuestro caso se usó el ordenamineto QuickSort.
Se podría decir que a mayor número de documentos auxiliares el ordenamineto se hará en menos iteraciones, pero consumirá más memoria y puede que tardé más que solo tener 2 archivos de apoyo.

Mezcla Directa

El algoritmo requiere de dos arreglos adicionales al arreglo que se está ordenando , "uno para hacer la cuenta" de los elementos y uno para entregar la salida ordenada.

La complejidad del algoritmo es (n+k) El compromiso "tiempo-espacio", se ve afectado ya que se resuelve en una menor cantidad de tiempo pero utiliza una gran cantidad de memoria la cual puede ser considerable para arreglos muy grandes. Resuminedo, parte el documento hasta ya no poder hacerlo divisible, y despues de esto, lo empieza a unir/fusionar hasta llegar al tamaño original del archivo, pero este ya ordenado.

Conclusiones

Calzada Martínez Jonathan Omar

Fue un trabajo pesado, sí no sabiamos por donde empezar, pero para nuestra fortuna ya habiamos hecho varios programas similares y solamente debiamos de transcribir hasta que funcionasé. Fue difícil porque este trabajo nos costó demasiadas horas, tuvimos la suerte de que teniamos horas libres para poder avanzar y no dividirnos el trabajo, sí lo hubieramos hecho cada quien por su cuenta, no hubiera entendido varias cosas. Es uno de los trabajos más extenuantes y gratificantes que he tenido últimamente.

García Lazcano Carlos David

La traducción del código no fue sencilla, puesto que se pasamos de ordenar números a ordenar Strings, además de leer y escribir en archivos auxiliares, fue un gran reto, y sinento que se logro el objetivo de una manera eficiente y sin salirnos tanto de lo ya visto en clase; añadiendo tambien que el nivel de dificultad del proyecto estuvo por encima de nuestro nivel hasta entonces.Lo nuevo que me llevo es poder usar y experimentar más con la utilería de java.io, puesto que en un futuro no muy lejano me enfrentaré a estos problemas, pero con mucha más información que se necesitara ordenar de acuerdo a los requerimeintos que se soliciten. Sin duda alguna me siemto satisfecho con el trabajo y sobre todo con el trabajo de equipo, que a pesar de llegar a momentos donde ya no sabiiamos que hacer, no nos dimos por vencido.

Jiménez Hernández Juan Carlos

Empezamos a modificar arcchivos que ya teniamos para que funcionaran con Strings, debido a que ya teniamos los códigos, y una vez que vimos que sí funcionaban procedimos a añadir la creacion de archivos, como teniamos tiempo suficiente, decidimos crear archivos en bucles y guardarlos en determinadas carpetas, creo y qye esta fue la parte más difícil de todas y como empezamos con Radix, una vez que terminamos este, fue más fácil trabajar con los otros dos metodos, a mi parecer el que da el ordenamineto más rápido es el de Distribución (Radix) y el que causa más problemas, al menos a mi punto de vista fue Polifase, debido a que se necesitan crear los archivos que el usuario quiera y de estas usar las iteraciones, fue un gran reto para terminarlo, y sientoq ue de no habernos dado 5 días más para terminarlo no lo hubiesemos entregado completo.

Bibliografía

Class BufferedReader
https://docs.oracle.com/javase/7/docs/api/

Class File
https://docs.oracle.com/javase/7/docs/api/
Polyphase
http://illuminatus.bizhat.com/algoritmos/polyphase.htm