Unidad 9

ARRAYS: Algoritmos de Ordenación y Búsqueda

Objetivos

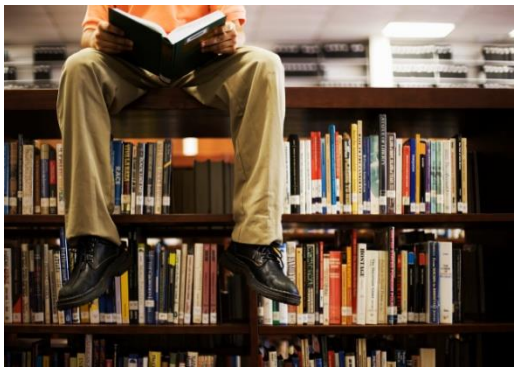
- Entender la importancia de la clasificación de datos
- Aprender y saber usar alguna técnica directa de clasificación de arrays.
- Aprender y saber usar algún algoritmo avanzado de ordenación de arrays.
- Uso de la clasificación para introducir conceptos sobre eficiencia de los algoritmos.
- Entender y saber explicar algunas formas de medir y comparar la eficiencia de distintos algoritmos de ordenación.
- Aprender distintas formas de búsqueda de elementos en un array.
- Aprender a insertar elementos en arrays ordenados o no.

Contenidos

- INTRODUCCIÓN
- CLASIFICACIÓN: CONCEPTO
  - Tipos de clasificación
- CLASIFICACIÓN INTERNA: MÉTODOS SENCILLOS
  - Método de Intercambio directo o de la Burbuja
- CLASIFICACIÓN INTERNA: MÉTODOS AVANZADOS
  - Método de Ordenación rápida o Quick Sort
- EFICIENCIA DE LOS ALGORITMOS
- BÚSQUEDA
  - Búsqueda secuencial o lineal
  - Búsqueda binaria
- INSERCIÓN

Introducción

- **Objetivo: mantener una estructura de datos ordenada para agilizar y hacer eficiente la búsqueda de elementos en dicha estructura.**



Concepto de clasificación

- **Clasificar significa establecer un orden entre los elementos que forman una estructura de datos según un determinado criterio.**
- **La estructura sobre la que se ordena o el criterio de clasificación pueden cambiar, sin que por ello varíe la regla que se aplica para ordenar.**

Tipos de clasificación

- **Según el orden**.
  - **Ascendentemente**: Los elementos están organizados de menor a mayor.
  - **Descendentemente**: Los elementos están organizados de mayor a menor.
- **Según la ubicación de los datos.**
  - **Interna:** Los datos se encuentran en memoria principal ⇨ ***Métodos de ordenación de arrays***
  - **Externa:** Los datos se encuentran en memoria secundaria ⇨ ***Métodos de ordenación de ficheros***
  - **Híbrida:** Los datos se encuentran en memoria secundaria, pero se ordenan en memoria principal volcándolos sobre una estructura de datos array**.**

Clasificación interna: métodos sencillos

Todos los métodos dividen el array en una parte ordenada y otra no ordenada.

- **INTERCAMBIO DIRECTO ó método de la BURBUJA**
  - **En cada iteración el elemento más pequeño de la parte no ordenada va burbujeando hasta llegar a su lugar correcto.** **algoritmo Pag. 4**

Realizar la traza con elementos repetidos y de forma descendente

Ejemplo Pag. 5

Videos

- http://www.neoteo.com/algoritmos-convertidos-en-danza

Clasificación interna: métodos avanzados

- **Ordenación Rápida o QUICK SORT**
  - **Basado en el de selección directa.**
  - **Se usa generalmente el enfoque recursivo por ser más sencillo que el iterativo.**
  - **Explicación:**
    - **Ordena un array, desde la primera casilla (*Primero*) hasta la última (*Ultimo*), siendo *PuntoParticion* la casilla por la que se divide el array. *PuntoParticion* contiene un elemento que está en su sitio, ordenado respecto a la colección de elementos del array.**
    - **Caso general: si Primero < Ultimo**
      - **Corta array por un punto, ordena la mitad izquierda y ordena la mitad derecha.**
    - **Caso base: si Primero >= Ultimo**
      - **No hace nada y marca el fin del procedimiento.**

Ver ejemplo página 9

Consideraciones sobre eficiencia

- **Es bueno evaluar algunos aspectos de los algoritmos (no únicamente los de ordenación), independientes de la plataforma o lenguaje en el que se vayan a implementar.**
  - **Estabilidad: Cómo se comporta con elementos iguales.**
    - **Algunos algoritmos mantienen el orden relativo entre éstos elementos y otros no.**
    - **Ej. Si tenemos la siguiente lista de datos (nombre, edad):**

      "Pedro 19, Juan 23, Felipe 15, Marcela 20, Juan 18, Marcela 17"*, y ordenamos alfabéticamente por el *nombre

      con un algoritmo estable quedaría así: *"Felipe 15, Marcela 20, Marcela 17, Juan 23, Juan 18, Pedro 19"*.

      Un algoritmo no estable podría dejar a Juan 18 antes de Juan 23, o a Marcela 20 después de Marcela 17.

Consideraciones sobre eficiencia

- **Requerimientos de memoria:** El algoritmo puede necesitar memoria adicional para realizar su tarea.
- **Tiempo de ejecución.**
- **Facilidad de implementación:** menos importante.

Cálculo de la eficiencia

- **Existen dos vías de cálculo:**
  - **Implementarlos, ejecutar y medir el tiempo de ejecución.**
    - **Depende de las condiciones de prueba: equipo, S.O., carga de trabajo adicional,...**
    - **Depende del lenguaje de programación que se use.**
    - **Hay que probar el algoritmo cuando el array está totalmente ordenado, parcialmente ordenado o totalmente desordenado.**
    - **Probar cada algoritmo para distintos tipos de datos y distintos tamaños del array.**
  - **Calcular analíticamente la eficiencia.**
    - **Evaluar parámetros como: Cantidad de bucles e iteraciones, de comparaciones, número de intercambios, cantidad de elementos.**

Orden de eficiencia: **METODOS DIRECTOS: N^2**

MÉTODOS AVANZADOS: N log N

Afectados por una Constante de proporcionalidad

Intercambio directo

- **Estabilidad: No intercambia casillas con valores iguales. Es *estable*.**
- **Requerimientos de Memoria: Sólo requiere de una variable adicional para realizar los intercambios, o memoria para las llamadas al módulo de intercambiar.**
- **Tiempo de Ejecución: El bucle interno se ejecuta n veces para un array de n elementos. El bucle externo también se ejecuta n veces. Este algoritmo presenta un comportamiento constante independiente del orden de los datos. Es decir, la complejidad promedia es del orden de $n * n = O(n^2)$.**

Intercambio directo

- **Ventajas:**
  - **Fácil implementación.**
  - **No requiere memoria adicional.**
  - **Rendimiento constante: poca diferencia entre el peor y el mejor caso.**
- **Desventajas:**
  - **Muy lento (es uno de los de más bajo rendimiento).**
  - **Realiza muchas comparaciones.**
  - **Realiza muchos intercambios.**

Quicksort

- **Estabilidad**: ***No es estable***.
- **Requerimientos de Memoria**: **No requiere memoria adicional en su forma recursiva.**
- **Tiempo de Ejecución**:
  - **La complejidad promedia es $O(n \log_2 n)$.**
- **Ventajas**:
  - **Muy rápido**
  - **No requiere memoria adicional.**
- **Desventajas**:
  - **Implementación un poco más complicada.**
  - **Recursividad (utiliza muchos recursos).**
  - **Mucha diferencia entre el peor y el mejor caso.**
- **Es un algoritmo muy eficiente. En general, será la mejor opción (salvo si se trata de ordenar pocos elementos)**

Otras consideraciones

- **Eliminación de las llamadas a subprogramas.**
  - Puede ser aconsejable sacrificar legibilidad para aumentar eficiencia, especialmente en arrays con poco número de elementos. Por ejemplo, eliminación del subprograma ***intercambiar***
- **Tiempo invertido por el programador**
  - Por ejemplo, los recursos físicos de un programa recursivo son más elevados que su versión iterativa, por lo que podría plantearse la implementación iterativa del QuickSort. Sin embargo, las horas que invertiría el programador en realizar esta última versión sería considerable al tratarse de una solución muy compleja. Por otro lado, el hardware es cada vez más rápido y económico, de manera que lo que resultaría más caro sería el tiempo del programador.
- **Ocupación de memoria.**
  - Esta consideración puede ser decisiva en algunos algoritmos. La ocupación es del orden de N, pero en el de **Mezcla** (Merge) es 2N.

Cuadro resumen

	Mejor Caso	Peor Caso	Caso Promedio	Estabilidad	Memoria Adicional
Burbuja	$O(n)$	$O(n^2)$	$O(n^2)$	Estable	No
Selección	$O(n^2)$	$O(n^2)$	$O(n^2)$	No Estable	No
Inserción	$O(n)$	$O(n^2)$	$O(n^2)$	Estable	No
Shell	$O(n \log n)$	Depende (*)	Depende (*)	No Estable	No
MergeSort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	Estable	Si
QuickSort	$O(n \log n)$	$O(n^2)$	$O(n \log n)$	No Estable	No

Tabla Resumen de los Algoritmos de Ordenación

(*): Depende de cómo se elijan los decrementos que se aplican en el Algoritmo.

Elegir el método más adecuado

- **No existe *EL* algoritmo de ordenación, sólo el mejor para cada caso.**
- **Debe conocerse bien el problema a resolver, y aplicar el más adecuado.**
- **Hay algunas preguntas que pueden ayudar a elegir:**
  - ***¿La información que se va a manejar está regular, muy o poco ordenada?***
  - ***¿Qué cantidad de datos se va a manipular?***
  - ***¿Qué tamaño tienen las casillas del array?***

Elegir el método más adecuado

- ***¿Cómo está la información a procesar?***
  - **Si la información está casi ordenada, un algoritmo sencillo como la burbuja será suficiente.**
  - **Si los datos van a estar muy desordenados, un algoritmo potente como *Quicksor*t puede ser el más indicado.**
  - **Si no se puede hacer una estimación sobre el grado de orden de la información, lo mejor será elegir un algoritmo que se comporte de manera similar en cualquiera de los casos extremos.**

Elegir el método más adecuado

- ***¿Qué cantidad de datos se va a manipular?***
  - **Si la cantidad es pequeña, es preferible uno de fácil implementación.**
  - **Una cantidad muy grande puede hacer prohibitivo usar un algoritmo que requiera mucha *memoria adicional.***
- ***¿Qué tamaño tienen las casillas del array?***
  - **Algunos algoritmos realizan múltiples intercambios (burbuja). Si las casillas tienen datos de gran tamaño estos intercambios son más lentos.**

Búsqueda

- **Búsqueda Secuencial ó Lineal**.
  - Recorre el array casilla a casilla hasta encontrar el elemento buscado o acabar el array
  - Es aplicable a arrays ordenados y desordenados.
- **Búsqueda Binaria.**
  - **Sólo para arrays ordenados**.
  - Explicación:
    - Divide el array en dos mitades y como el array está ordenado busca en la parte donde sea previsible que esté el elemento buscado.
    - Acaba el proceso cuando se detecte que el dato no está en el conjunto de elementos del array o bien sea encontrado.

Inserción

- **Distintas situaciones para insertar un elemento en un array:**
  - **El array puede o no estar lleno.**
  - **El elemento a insertar puede o no estar ya en el array.**
  - **Puede o no permitirse elementos repetidos.**
  - **El array puede o no estar ordenado.**
  - **Debe informarse del éxito o fracaso de la inserción.**

"Ordenar bibliotecas es ejercer de un modo silencioso el arte de la crítica".

Jorge Luís Borges (1899 -1986); Escritor