UT9 – Clasificación

Una técnica importante de diseño de algoritmos es usar clasificación como componente base para:

* Búsqueda (ej, búsqueda binaria)
* Par más cercano
* Unicidad de elementos
* Distribución de frecuencias (¿qué elemento ocurre la mayor cantidad de veces?)
* Selección (¿cuál es el k-ésimo mayor elemento?)

Los algoritmos más simples requieren tiempos de O(n2 ), otros O(n \* log n) y algunos, para clases especiales de datos, O(n).

Los buenos algoritmos de clasificación requieren O(n\*log n) comparaciones, los más sencillos, O(n2 ) •

Métodos directos e indirectos. – Los directos son más cortos y fáciles de entender – Las operaciones de los indirectos son más complejas, por lo que los métodos directos pueden ser más rápidos para pequeños conjuntos de datos

Partes > Interna (se produce en la memoria principal donde es posible aprovechar la capacidad de acceso aleatorio)

* Externa (es necesaria cuando un número de objetos a ordenar es demasiado grande para entrar en memoria principal)

Principales algoritmos de clasificación interna: (los algoritmos más simples requieren O(n2))

* Quicksort lleva un tiempo de O(n log n), peor caso de O(n2).
* Heapsort (clasificación por montículos) y mergesort (por intercalación) llevan O(n log n) en el peor de los casos, aunque en el promedio de casos peor que Quicksort.
* Clasificación por cubetes (o urnas). Opera solo con clases especiales de datos, pero requieren tiempo O(n) en el peor caso.

Criterios para evaluar un algoritmo de clasificación -> contar el número de pasos requeridos por el algoritmo para clasificar n registros.

-> el número de comparaciones entre claves para clasificar n registros.

-> si el tamaño del registro es grande, conviene contar las veces que debe moverse.

**Clasificación de Burbuja:**

Imaginar que los registros a almacenar están en un arreglo vertical, los registros livianos (menores) suben. El menor sube hasta su punto más alto, si ya subió uno antes, este queda atrás de él.

**Clasificación por inserción:**

En el i-ésimo recorrido se inserta el i-ésimo elemento A[i] en el lugar correcto entre A[1], A[2], …, A[i – 1] los cuales fueron ordenados previamente. El primer valor debe ser lo mínimo posible, para finalizar el proceso de mover A[i].

A[0] = -∞

for i = 2 a n hacer

j = i

while A[j] < A[j – 1] do

intercambia (A[S], A[j – 1])

j = j – 1

end

end

**Clasificación por inserción:**

En el i-ésimo recorrido se selecciona el registro con la clave más pequeña entre A[j] … A[n] y se intercambia con A[i]. Como resultado después de i pasadas, los i registros menores ocuparan A[1] … A[i], en el orden clasificado.

For i=1 to n-1 do

seleccionar el más pequeño entre A[i], …, A[n] e intercambiarlo con A[i]

Las clasificaciones de burbuja, por inserción y por selección llevan O(n^2) y Ω(n^2).

A menos que n sea aproximadamente 100 puede ser una pérdida de tiempo implementar algoritmos muy complicados.

**Clasificación de Shell (generalización de la clasificación de burbuja)**

Es O(n^1.26)

**Clasificación rápida (quicksort)**

Consiste en clasificar arreglo A[1], …, A[n] tomando un valor como elemento pivote (cercano a la mitad) alrededor del cual reorganizar los elementos del arreglo.

Se permutan los elementos con el fin de que, para alguna j, todos los elementos con claves menores aparezcan A[1], … , A[j] y todos con claves mayores aparezcan A[j+1], … , A[n]. Despues se aplica recursivamente.

**Clasificación Heapsort (clasificación por monticulos):**

Peor caso y uso promedio (usando árbol) O(nlogn).

Este algoritmo puede representarse de forma abstracta por medio de las 4 operaciones de conjuntos: Insertar, suprimir, vacía y min

**Clasificación por urnas (binsort):**

Un proceso de clasificación donde se crea una urna para contener todos los registros con cierta clave. Se examina cada registro r a clasificar y se coloca en la urna de acuerdo con el valor de la clave de r.