

CI2612: Algoritmos y Estructuras de Datos II

Blai Bonet

Universidad Simón Bolívar, Caracas, Venezuela

Problema de ordenamiento y cálculo de estadísticos

© 2018 Blai Bonet

Objetivos

- Introducir los problemas de ordenamiento y cálculo de estadísticos
- Discutir su relevancia y resultados

© 2018 Blai Bonet

Ordenamiento

Dada una secuencia $\{a_1, a_2, \dots, a_n\}$ de n números, un algoritmo de ordenamiento produce una permutación (reordenamiento) $\langle a'_1, a'_2, \dots, a'_n \rangle$ de la entrada tal que $a'_1 \leq a'_2 \leq \dots \leq a'_n$

Sin embargo, lo que frecuentemente ordenamos son **secuencias de registros** donde cada registro tiene una **clave** asociada, tomada de un universo U

Los datos que acompañan a la clave en un registro se llaman **datos satélite**

El universo U es **totalmente ordenado**: dadas claves k_1 y k_2 en U , uno y sólo uno de los siguientes se cumple $k_1 < k_2$, $k_1 = k_2$, o $k_2 < k_1$

Aunque presentamos algoritmos de ordenamiento para números, un algoritmo de ordenamiento debe lidiar con los datos satélite

© 2018 Blai Bonet

Propiedades de algoritmos de ordenamiento

Un algoritmo de ordenamiento es:

- **“in place”**: si a lo sumo un número constante de elementos de la entrada son guardados fuera del arreglo de entrada durante la ejecución
- **estable**: si dos elementos a_i y a_j que sean iguales aparecen en la salida en el mismo orden relativo al que aparecen en la entrada (i.e. si $a_i = a_j$ e $i < j$, entonces los elementos $a'_{i'}$ y $a'_{j'}$ en la salida que refieren a a_i y a_j son tales que $i' < j'$)

Importancia del problema de ordenamiento

- Un arreglo ordenado permite hacer **búsqueda binaria**
- Varios algoritmos fundamentales requieren hacer algún tipo de ordenamiento dentro de su ejecución
- Ordenamiento también se utiliza para remover duplicados y para construir formas canónicas de objetos
- ...

Comparación de algoritmos de ordenamiento

Algoritmo	Tiempo en el peor caso	Tiempo promedio/esperado	Propiedades estable/in-place
Insertion-Sort	$\Theta(n^2)$	$\Theta(n^2)$	estable/in-place
Mergesort	$\Theta(n \log n)$	$\Theta(n \log n)$	estable
Heapsort	$\Theta(n \log n)$	$\Theta(n \log n)$	in-place
Quicksort	$\Theta(n^2)$	(promedio) $\Theta(n \log n)$	in-place
Randomized-Quicksort	$\Theta(n^2)$	(esperado) $\Theta(n \log n)$	in-place
Counting-Sort	$\Theta(k + n)$	$\Theta(k + n)$	estable
Radix-Sort	$\Theta(d(n + k))$	$\Theta(d(n + k))$	estable
Bucket-Sort	$\Theta(n^2)$	(promedio) $\Theta(n)$	estable

Cálculo de estadísticos de orden

El i -ésimo estadístico de orden de un conjunto de n elementos es el i -ésimo menor elemento del conjunto

Ordenando el conjunto podemos seleccionar el i -ésimo estadístico en tiempo $O(n \log n)$

Veremos dos algoritmos eficientes para calcular estadísticos:

- **algoritmo randomizado** que corre en tiempo esperado $O(n)$
- **algoritmo determinístico** que corre en tiempo $O(n)$ en el peor caso