Estructuras de Datos y Algoritmos – IIC2133

I1

4 abril 2022

Inicio: 18.30
Entrega: hasta las **20.30**

Responde sólo 3 de las siguientes 4 preguntas

1) Ordenar por dos criterios.

Como parte del proceso de postulación a las universidades chilenas que realiza el DEMRE, se cuenta con un gran volumen datos en que cada registro representa cada una de las postulaciones de cada estudiante a una carrera en una universidad. Estos registros son de la siguiente forma:

RUT_Postulante	codigo_universidad	codigo_carrera	puntaje_ponderado	…

Nota: Asume que codigo_universidad y codigo_carrera son valores numéricos enteros.

Originalmente las postulaciones se encuentran ordenadas por RUT_Postulante, y se requiere ordenarlas por codigo_universidad Y codigo_carrera para distribuir esta información por separado a cada universidad con los postulantes a cada una de sus carreras.

a) Propón un algoritmo para realizar el ordenamiento requerido. Puedes utilizar listas o arreglos —especifica. Usa una notación similar a la usada en clases.
b) Calcula su complejidad.
c) Determina su mejor y peor caso.

Hint: Puedes referirse al valor del atributo de un elemento como *elemento.atributo*, por ejemplo, si P es uno de los registros de postulaciones, entonces:

P.*codigo_universidad* retorna el valor del codigo_universidad de la postulación
P.*codigo_carrera* retorna el valor del codigo_carrera de la postulación

2) Ordenación estable.

Un algoritmo de ordenación es **estable** si al ordenar dos elementos con el mismo valor los deja en el mismo orden relativo que tenían antes de ordenarlos. Por ejemplo, considera el siguiente arreglo de nombres de personas y las edades correspondientes:

[(yo, 23), (tú, 20), (él, 2), (ella, 10), (nosotros, 25), (vosotros, 15), (ellos, 25), (ellas, 30), (ustedes, 10), (vosotras, 5), (todos, 12) **]**

Si al ordenar este arreglo por edades, *ella* queda antes que *ustedes* y *nosotros* antes que *ellos*, entonces el algoritmo de ordenación es estable; en otro caso, inestable.

a) Ejecuta el algoritmo *selection sort* estudiado en clases sobre el arreglo anterior y muestra que es estable.
b) Sugiere una forma de hacer que *selection sort* sea inestable, pero mantenga su propiedad de ordenar.

3) Estrategias algorítmicas. Sean X e Y conjuntos de datos **no** ordenados. Lo que se busca hacer es lo siguiente: Se calcula el producto cartesiano entre ambos conjuntos; o sea, todos los pares posibles con un elemento de X y otro de Y. Sea Z este nuevo conjunto.

Formalmente, producto cartesiano $Z = \{(x_i, y_i) \mid x_i \in X, y_i \in Y\}$.

Se pide ordenar dichos pares según el valor de la suma $z = x + y$. Y en caso de empates, dejar primero al que estaba primero con respecto a x. Asume que cada suma en Z es única.

a) Describe una forma de calcular las sumas de forma eficiente en un entorno altamente paralelizado
b) Dadas las sumas, describe una estructura que permita mantener los pares ordenados y la suma correspondiente
c) Finalmente, describe un algoritmo de ordenación que ordene eficientemente los pares ordenados basándose en el resultado de la suma

Hint: Recuerda estrategias algorítmicas vistas en clases.

4) Demostración de corrección. Considera dos arreglos A y B de n elementos cada uno; ambos arreglos están ordenados. Demuestra que el siguiente algoritmo encuentra la mediana de la totalidad de los elementos de A U B en O(log n).

1) Calcula las medianas *m1* y *m2* de los arreglos ar1[] y ar2[] respectivamente.
2) If *m1* = *m2* → terminamos; return *m1* (o *m2*)
3) If *m1* > *m2*, entonces la mediana está en uno de los siguientes subarreglos:
   a) Desde el primer elemento de ar1 hasta *m1*
   b) Desde *m2* hasta el último elemento de ar2
4) If *m2* > *m1*, entonces la mediana está en uno de los siguientes subarreglos:
   a) Desde *m1* hasta el último elemento de ar1
   b) Desde el primer elemento de ar2 hasta *m2*
5) Repite los pasos anteriores hasta que el tamaño de ambos subarreglos sea 2.
6) If tamaño de ambos subarreglos es 2, entonces la mediana es:
   Mediana = (max(ar1[0], ar2[0]) + min(ar1[1], ar2[1]))/2

Específicamente, demuestra —es decir, da un argumento lógico, bien razonado— que

a) El algoritmo termina (observa que el algoritmo es iterativo, que en cada iteración el tamaño de los subarreglos disminuye, y que la condición de término de la iteración es que el tamaño de ambos subarreglos sea 2).
b) El algoritmo efectivamente encuentra la mediana en tiempo O(log n) (recuerda que dado que el número total de elementos es par, la mediana es el promedio aritmético de los dos elementos que quedan "al medio" si todos los elementos estuvieran ordenados). Con respecto a los pasos 3 y 4, basta que demuestres uno de los dos.