Más D&C, Más Sorting

Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

Algoritmos y Estructuras de Datos II

Ejercicio: Rankings

Se tiene un arreglo R con n strings sin repeticiones que define un *ranking*.
Contamos, además, con un arreglo A de m strings tal que todos ellos aparecen en el ranking R
Se quiere ordenar el arreglo A en función del ranking definido por R.
Por ejemplo, si tenemos

$R =$ `[Brasil, Argentina, Alemania, Chile, Colombia, Francia]`

$A =$ `[Chile, Francia, Brasil, Chile, Argentina, Brasil]`

entonces el orden correcto para A sería

`[Brasil, Brasil, Argentina, Chile, Chile, Francia]`

Suponiendo que el largo de todos los strings está acotado por una constante, proponer un algoritmo de ordenamiento que resuelva el problema en una complejidad $O(n + m)$, donde n y m son las cantidades de elementos en el ranking y en el arreglo a ordenar, respectivamente.

Ejercicio: Secuencias de intervalos

Se tiene un arreglo de secuencias no vacías de números naturales, cada una de los cuales contiene a todos los números dentro de un intervalo determinado. Se quiere generar un arreglo de naturales que contenga a todos los elementos de todas las secuencias, sin repetidos, ordenados de menor a mayor.

entrada	[[8 5 7 6] [11 10 12] [4 3 5 6]]
salida	[3 4 5 6 7 8 10 11 12]

Proponer un algoritmo que resuelva el problema en $O(n + (k \log k))$, donde k es la cantidad de secuencias y n la cantidad total de elementos (es decir, la suma de las longitudes de todas las secuencias).

Ejercicio: Intersección más grande

Dado un arreglo de $n \geq 2$ intervalos cerrados de números naturales $I_1, \ldots, I_n$ (cada uno representado como un par $< l_{inf}, l_{sup} >$) ordenado[1], se desea encontrar dos de ellos que maximicen su intersección, es decir, un par de índices i y j con $1 \leq i < j \leq n$ tal que la cantidad de valores dentro de la intersección de los intervalos $I_i \cap I_j$ sea máxima entre todos los intervalos de entrada.

Dar un algoritmo que use la técnica de *Divide and Conquer* y resuelva el problema en tiempo $O(n \log n)$ en el peor caso.

- Marcar claramente qué partes del algoritmo se corresponden a *dividir*, *conquistar* y *unir* subproblemas.
- Justificar detalladamente que el algoritmo cumple con la complejidad pedida.

[1] usando l_{inf} como clave

Ejercicio: Árbol bicolor

Un árbol binario bicolor[2] es *rojinegro válido*[3] si y sólo si satisface las siguientes 3 propiedades:

- Todas las hojas son negras.
- Los hijos de un nodo rojo, de haberlos, siempre son negros.
- Todos los caminos (desde la raíz hasta una hoja) contienen la misma cantidad de nodos negros.

Escriba un algoritmo *EsRojinegroVálido?*, basado en la técnica de dividir y conquistar, que dado un árbol binario bicolor determine si es rojinegro válido.
No visite el mismo nodo múltiples veces. Marque claramente en su algoritmo las distintas etapas de la técnica.
Suponga que puede determinar en O(1) si un nodo es rojo o negro.
Calcule la complejidad temporal de su algoritmo en peor caso.

---

[2] Un árbol es bicolor si sus nodos son de color rojo o negro.

[3] La definición que usamos en este ejercicio es una versión simplificada de los *red-black trees* clásicos.