CC4102 - Control 2

Profs. Pablo Barceló y Gonzalo Navarro

13 de Mayo de 2019

P1 (2.0 pt)

Se propone la siguiente solución intermedia sencilla entre tener n elementos ordenados en un array, donde una inserción cuesta $O(n)$ y una búsqueda $O(\log n)$, o tenerlos en una lista enlazada, donde la búsqueda cuesta $O(n)$ pero la inserción cuesta $O(1)$ (si se sabe dónde insertar).

Tenemos $\log_2 n$ arrays $A_0, A_1, \ldots$. El array A_i está vacío o tiene exactamente 2^i elementos ordenados. Para buscar, se busca en cada uno de los arrays. Para insertar x, se intenta meter en A_0. Si está vacío, terminamos. Si está lleno, hacemos el merge de x con A_0 y movemos los dos elementos a A_1. Si A_1 estaba vacío, terminamos. Si está lleno, hacemos el merge con A_1 y movemos los cuatro elementos a A_2, y así hasta encontrar un arreglo vacío.

1. Indique el peor caso del tiempo de búsqueda y de inserción. Muestre que sus cotas son ajustadas.
2. Indique el costo amortizado de inserción partiendo de un arreglo vacío, y demuestre que no podría ser menor.

P2 (2.0 pt)

El vEB tree almacena n elementos de un universo $[0..u-1]$, en espacio $O(u)$ y tiempo de búsqueda del predecesor $O(\log \log u)$. Para mejorar su desempeño en conjuntos densos (n cercano a u), se propone cortar el universo en n subuniversos de tamaño u/n, guardando un vEB para cada subuniverso. Explique cómo encontrar el predecesor en tiempo $O(\log \log(u/n))$ con esta estructura, y calcule el espacio resultante. Recuerde que siempre puede usar $O(n)$ espacio adicional para otras estructuras.

P3 (2.0 pt)

Se tiene el árbol de sufijos para un texto $T[1..n]$, y un nuevo string $S[1..m]$, sobre un alfabeto constante. Se quieren determinar los substrings maximales de S que aparecen en T, es decir todos los $S[i..j]$ que aparecen en T pero ni $S[i-1..j]$ ni $S[i..j+1]$ aparecen en T.

Diseñe un algoritmo de tiempo $O(m)$ para resolver este problema. Por simplicidad, considere que tiene el trie de los sufijos de T, más que el árbol de sufijos. Recuerde que cada nodo que representa el string aX tiene un *suffix link* que lleva al nodo que representa X, donde a es un símbolo y X un string.

Tiempo: 2.0 horas | Con una hoja de apuntes | Responder en hojas separadas