Auxiliar 13 - Auxiliar Sandwich

CC4102 - Diseño y Análisis de Algoritmos
Profesor: Gonzalo Navarro Auxiliar: Jorge Bahamonde

7 de Diciembre del 2015

1 Árboles α-balanceados

Un *árbol α-balanceado*, para $1/2 < \alpha < 1$, es un árbol binario de búsqueda donde todo subárbol $T = (root, T_l, T_r)$, cumple $|T_l| \leq \alpha|T|$ y $|T_r| \leq \alpha|T|$. Las operaciones para buscar y mantener un árbol α-balanceado son las mismas que para un árbol binario de búsqueda, excepto que luego de insertar o borrar un nodo, se busca el nodo más alto en el camino del punto de inserción/borrado hacia la raíz, que no esté α-balanceado, y se lo reconstruye como árbol perfectamente balanceado (el costo es proporcional al tamaño del subárbol que se reconstruye).

1. Muestre que la búsqueda en un árbol α-balanceado cuesta $O(\log n)$, y que lo mismo ocurre con las inserciones y borrados, si no consideramos las reconstrucciones. ¿Qué constante obtiene multiplicando el $\log n$?

2. Muestre que el costo amortizado de las inserciones y borrados, ahora considerando las reconstrucciones, es también $O(\log n)$. Para ello, considere la función potencial

$$\Phi(T) = \frac{1}{2\alpha - 1} \sum_{T' \in T} \max\{||T'_l| - |T'_r|| - 1, 0\}$$

   donde $T' \in T$ significa que T' es un subárbol de T.

2 Dominios... ¿Discretos?

Se desea ordenar n puntos que pertenecen al círculo unitario según su distancia al origen, de menor a mayor. Diseñe un algoritmo que en promedio tome tiempo $O(n)$, suponiendo que los puntos se distribuyen de manera uniforme en este espacio.

3 Coloreo de Elementos

Sea S un conjunto de n elementos. Se tienen k conjuntos $S_1, \ldots, S_k$ de r elementos cada uno. Los conjuntos no necesariamente son disjuntos, y se cumple que $k/2^r \leq 1/4$. Se desea colorear los elementos de rojo o azul, de modo que ningún S_i quede monocromático (todos los puntos rojos o todos azules) .

1. Diseñe un algoritmo de tipo MonteCarlo que obtenga un coloreo válido con probabilidad $1/2$ y analícelo.

2. Mejore su algoritmo para obtener un coloreo válido con probabilidad de fallar a lo más $1/2^t$, para cualquier t dado, y analícelo.

3. Convierta el algoritmo en uno de tipo las Vegas y analice su costo esperado.

4 Uniendo (casi) todo: Heavy Hitters

Considere un *stream* A de elementos $a_1, ..., a_m$ de m objetos, donde cada $a_i \in 1, ..., n$. Esto significa que el tamaño de cada elemento es cercano a $\log n$, y que contar el número de objetos ya vistos requiere espacio $\log m$ (en el caso exacto). En este caso, n puede ser *muy* grande.

Denotaremos $f_j = |\{a_i \in A \mid a_i = j\}|$, es decir, el número de elementos en el stream que toman el valor j. También denotaremos $F_1 = \sum_j f_j$, el número de elementos ya vistos, y $F_0 = \sum_j f_j^0$ el número de elementos distintos.

Queremos resolver el problema de encontrar aquellos elementos con una frecuencia mayor a ϕ, es decir, que $f_j > \phi \cdot m$ (análogamente, el problema puede ser planteado como encontrar los k elementos más frecuentes).

Analizaremos una versión aproximada del problema, de modo que se permite que elementos con una frecuencia entre $\phi - \epsilon$ y ϕ también sean retornados.

Resolveremos este problema con un algoritmo online y probabilístico que entrega una solución aproximada. El análisis requerirá pensar,además, en las representaciones de los elementos.