Auxiliar 4 - Análisis Amortizado

CC4102 - Diseño y Análisis de Algoritmos
Profesor: Pablo Barceló Auxiliar: Jorge Bahamonde

17 de Abril del 2015

1. La búsqueda binaria en un arreglo ordenado toma tiempo logarítmico, sin embargo la inserción toma tiempo lineal. Veremos que podemos mejorar el tiempo de inserción manteniendo varios arreglos ordenados.

   Específicamente, suponga que se desea implementar las operaciones de búsqueda e inserción en un conjunto de n elementos. Sea $k = \lceil \log(n+1) \rceil$ y asuma que la representación binaria de n es $n_{k-1} \cdots n_1 n_0$. Utilizaremos k arreglos ordenados $A_0, \cdots, A_{k-1}$, donde cada A_i tiene largo 2^i. Cada arreglo está lleno o vacío dependiendo si $n_i = 1$ o $n_i = 0$, respectivamente. El número total de elementos contenidos en los k arreglos es entonces $\sum_{i=0}^{k-1} n_i 2^i = n$. Aunque cada arreglo A_i está ordenado, no existe ninguna relación particular entre los elementos de distintos arreglos.

   (a) Muestre como realizar la operación de búsqueda en esta estructura de datos. Analice el costo de peor caso.

   (b) Muestre como realizar la operación de inserción. Analice el costo de peor caso y el costo amortizado de insertar.

   (c) (Propuesto) Discuta como implementar la operación de eliminar un elemento. Cuál es el costo amortizado de operaciones insertar/borrar?

2. Considere una tabla de tamaño $size$ que almacena num elementos, que se va llenando con inserciones y debemos realocarla cuando no queda espacio para insertar ($size = num$). Esta vez, también ocurren borrados y no queremos que $size$ sea mucho mayor que num, para no desperdiciar espacio. Al realocar la tabla para agrandarla o reducirla debemos pagar un costo de $O(num)$.

   (a) Muestre que duplicar la tabla cuando se llena, y reducirla cuando $num = size/2$ no consigue un costo amortizado constante.

   (b) Considere la estrategia de duplicar cuando la tabla se llena, y reducirla a $size/2$ cuando $num = size/4$. Demuestre que esta estrategia obtiene un costo amortizado constante utilizando la siguiente función potencial:

   $$\Phi = \begin{cases} 2 \cdot num - size & \text{si } num \geq size/2 \\ size/2 - num & \text{si } num < size/2 \end{cases}$$

   (c) Analice el caso general en que queremos obtener un factor de carga α, con $0 < \alpha < 1/2$.

3. 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).

(a) 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 $\log n$?

(b) 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.