Auxiliar 6 - Análisis Amortizado

CC4102/CC53A - Diseño y Análisis de Algoritmos
Profesor: Pablo Barceló   Auxiliar: Miguel Romero

24 de Abril del 2014

1. Queremos implementar una *lista auto-organizada*, esto es, una lista enlazada desordenada en donde:

   - Cada vez que se busca una llave en la lista, recorremos secuencialmente hasta encontrarla. Si la llave a buscar está en la posición i, entonces el costo de esta operación es i.
   - Después de buscar un elemento, podemos reordenar los elementos en la lista, mediante una serie de *swapeos* entre elementos contiguos. Cada swapeo tiene costo 1.

   Luego el costo de buscar una llave k para una heurística H que implemente una lista auto-organizada, es (posición de k en la lista) + (cantidad de swapeos).

   Definimos la heurística *move-to-front* (MTF) como sigue: Después de buscar una llave k en la lista, ubicamos k al principio de la lista. Observe que el costo de MTF para buscar k es 2(posición de k en la lista)$-$1. Se pide probar que el costo amortizado de buscar para MTF es a lo más 4 veces el costo amortizado de buscar para *cualquier* heurística. Para esto demuestre que, si H es cierta heurstica, L_0 es cierta lista enlazada inicial, y $\sigma = (\sigma_1, \ldots, \sigma_m)$ es una secuencia de búsqueda, en donde cada σ_i es una llave en L_0, entonces $C_{MTF}(\sigma) \leq 4C_H(\sigma)$, donde $C_{MTF}(\sigma)$ y $C_H(\sigma)$ es el costo de la secuencia σ para MTF y H, respectivamente, cuando parten desde la lista L_0.

   Hint: Suponga que L_i es la lista después de las búsquedas $\sigma_1, \ldots \sigma_i$ usando MTF, y L_i^* usando H. Defina una *inversión* como un par de llaves (y, z) tal que y precede z en L_i, pero z precede y en L_i^*. Aplique el método de la función potencial, con la función $\Phi(L_i) = 2q_i$, donde q_i es el número de inversiones en L_i, con $0 \leq i \leq m$.

2. (C2 P1 2013/02) 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.