# Auxiliar 8 - Más análisis amortizado y dominios discretos

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

9 de Noviembre 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.

   Analice las operaciones de búsqueda e inserción en esta estructura.

2. Un *multistack* consiste en una serie (potencialmente infinita) de stacks $S_0, ..., S_{t-1}$ donde el j-ésimo stack puede almacenar hasta $3^j$ elementos. Todas las operaciones de *push* y *pop* se realizan inicialmente sobre $S_0$.Cuando se desea *pushear* un elemento en un stack lleno $S_j$, se vacía este stack en el siguiente, $S_{j+1}$ (posiblemente repitiéndose la operación de forma recursiva). Al hacer *pop* de un stack vacío,se llena éste a partir de *pops* del siguiente stack (posiblemente repitiéndose la operación de forma recursiva) y luego se hace el *pop* correspondiente. Considere que las operaciones de *push* y *pop* en los stacks individuales tiene costo 1.

   - En el peor caso, ¿cuánto cuesta una operación de *push* en esta estructura?
   - Demuestre que el costo amortizado de una secuencia de $n$ operaciones de *push* en un multistack inicialmente vacío es de $O(n \log n)$. Utilice la siguiente función de potencial:

     $$\Phi = 2 \sum_{j=0}^{t-1} N_j \cdot (\log_3 n - j)$$

     donde $N_j$ es el número de elementos en el j-ésimo stack.
   - Demuestre lo mismo para cualquier secuencia de *pushes* y *pops*.

3. Una operación de interés sobre cadenas binarias es SelectNext$(B, i)$, que retorna la posición del siguiente 1 después de la posición $i$. Usando una metodología similar a la usada para Rank (uso de bloques y superbloques) diseñe una estructura que requiera $o(n)$ bits extra y que permita responder SelectNext en tiempo constante. Note que este problema es similar al de encontrar el sucesor al elemento $i$.

4. Describa un algoritmo que, dados $n$ enteros en $[0, ..., k]$, preprocese su entrada y responda cuántos de estos enteros se encuentran en el rango $[a, ..., b]$ en tiempo constante. Su algoritmo debería tomar tiempo $\Theta(n + k)$ en el preprocesamiento.