CC4102/CC53A - Diseño y Análisis de Algoritmos
Auxiliar 3

Prof. Gonzalo Navarro; Aux. Mauricio Quezada

4 de Septiembre de 2012

1 Heavy Hitters (Ex. 2011/2)

El problema de los *heavy hitters* es, dada una secuencia, encontrar k elementos con frecuencia mayor a f, y además reportar esa frecuencia. Tenemos n elementos, pero éstos son demasiados para la capacidad de la memoria principal, M.

1. Diseñe y analice un algoritmo de memoria secundaria para encontrar k heavy hitters, suponiendo que tiene $M = O(k)$ espacio en memoria principal.

2. Suponga ahora que debe resolver el problema en modo *streaming*, es decir, sólo hay tiempo para leer los datos de disco una sola vez. Diseñe un algoritmo que encuentre heavy hitters tal vez cometiendo errores, pero con alguna clase de garantía.

2 Distribution Sort (T1 2011/2)

Diseñe un algoritmo de ordenamiento en memoria secundaria inspirándose en Quicksort, de manera similar a la adaptación de Mergesort en memoria secundaria y analice su costo de I/O.

3 R-Tree (C1 2011/2)

Un *R-Tree* es un árbol, parecido al B-Tree, que almacena un conjunto de rectángulos en memoria secundaria. Se permite insertar y borrar rectángulos del conjunto. La consulta que se realiza es: dado un rectángulo C, encontrar todos aquellos que tienen intersección con C.

Cada nodo interno del R-Tree tiene varios hijos, y para cada hijo almacena el mínimo rectángulo que contiene a todos los rectángulos almacenados en las hojas que descienden de ese hijo (se llama MBR: minimum bounding rectangle). En las hojas se almacenan los rectángulos reales. Para resolver una consulta C, se ve en la raíz cuáles MBRs intersecta C y se sigue recursivamente por esos hijos.

Con esta descripción sucinta y sin pedir más detalles:

1. Dé un pseudocódigo para el algoritmo de búsqueda. Explique por qué es mejor que los MBRs sean pequeños

2. Diseñe un algoritmo de inserción y borrado de rectángulos que requiera una cantidad logarítmica de I/Os. Discuta los pro y contra de las alternativas.