Streaming Problem Set*

Due: Wednesday, October 05, 2011
Turn in at the start of class.

1 Building B-Trees in External Memory

Recall a B-Tree is an (a, b)-Tree with $a, b = \Theta(B)$, and each node stored in a separate block in memory. A B^+-Tree is like a B-Tree, but all of the items are stored in the leaf blocks, in sorted order. Each internal node of the tree has some value $k = \Theta(B)$ children where $k \in (a, b)$; the node stores k pointers (one to each child) and $k + 1$ values $\{v_1, v_2, \ldots, v_{k_1}\}$ such that the ith child only has items with values between v_i and v_{i+1}.

How many I/Os does it take to build a B^+-Tree on an input of N items stored in $O(N/B)$ consecutive blocks? Describe the algorithm and prove the number of I/Os needed, you may site any results we proved in class.

2 Aggressive Approximate Counts

Consider the following streaming algorithm on a stream $A = \langle a_1, a_2, \ldots, a_m \rangle$ where each $a_j \in [n]$.

Keep a data structure S that consists of k counter-index pairs $\{(c_1, t_1), (c_2, t_2), \ldots, (c_k, t_k)\}$. Each $t_i \in [n]$ and labels the index in the universe of stream elements $[n]$. Each c_i keeps track of a count of that index t_i.

To process a new $a_j \in A$:

- **IF** for some i we are maintaining that index ($t_i = a_j$), then update $c_i = c_i + 1$.
- **ELSE IF** for some i the count is empty ($c_i = 0$), then set $t_i = a_j$ and $c_i = 1$.
- **ELSE** Let $c_{\min}$ be the count of the pair with the smallest count. Choose an arbitrary pair i with $c_i = c_{\min}$ and set $c_i = c_i + 1$ and $t_i = a_j$.

Let $f_x = |\{a_j \in A \mid x = a_j\}|$ is the count of elements in the stream with value x. Our data structure approximates the value f_x with a value $S(x)$ as follows. If there exists some i such that $t_i = x$, then return $S(x) = c_i$. Otherwise, return $S(x) = m/k$.

We want to bound the approximation factor of this algorithm in terms of m and k. How large can $|S(x) - f_x|$ be? We will build up to an answer through the following parts. Answer each one, and explain why. (The explanation why should typically be one or two sentences - extremely long explanations will be marked incorrect.)

Part A: If $k = 1$, what are the possible values that $S(x)$ can return?

Part B: If $k = 2$, what is the value of $c_1 + c_2$?

Part C: If $k = 2$, and some $f_x > m/2$, how small and can $S(x)$ be?

---

*CS 7960 Models of Computation for Massive Data

Part D: Let there be only k elements $X = \{x_1, \ldots, x_k\}$ such that $f_{x_i} > 0$. For $x_r \in X$, how large can $|S(x_r) - f_{x_r}|$ be?

Part E: After processing $m' < m$ items, consider a case where the algorithm processes some x and no $t_i = x$. After we reassign some pair $t_i = x$ and $c_i = c_i + 1$, how large can c_i be?

Part F: Assume some $t_i = x$. How large can $S(x) - f_x$ be? (*Hint: use Part E*)

Part G: Consider a case where the algorithm processes some x and no $t_i = x$. Assume we reassign some pair $t_i = x$ and $c_i = c_i + 1$, where $t_i = x'$ before this element was processed. Before this element was processed, can the stream have already processed more than c_i elements with index x'? (*Hint: Let y be the number of elements with index x' already processed. Use induction on y to bound c_i in terms of y where $t_i = x'$.*)

Part H: If some $t_i = x$, can $S(x)$ be smaller than f_x? (*Hint: use Part G*)

Part I: Can an $x \in [n]$ such that $f_x > m/k$ have no $t_i = x$ at the termination of the algorithm? (*Hint: Use Part G*)

Part J: How large can $|S(x) - f_x|$ be? (*Hint: Use results you proved in previous parts.*)

3 Randomized Approximate Range Searching

Consider the following streaming algorithm on a stream $A = \langle a_1, a_2, \ldots, a_m \rangle$ where each $a_j \in [n]$.

Keep a data structure S of k points maintained as follows. Maintain k independent reservoir samples of size 1. That is keep k indices $\{t_1, t_2, \ldots, t_k\}$. When processing a_j, for each $i \in [k]$ independently set $t_i = a_j$ with probability $1/j$, otherwise do not update t_i.

We want S to be able to answer size queries. Given a subset $R \subset [n]$, how large is $\mathsf{size}(R) = \sum_{x \in R} f_x$. Describe how to query $S(R)$ so that we can guarantee, with probability at least $1 - \delta$,

$$\mathsf{size}(R) - \varepsilon m \leq S(R) \leq \mathsf{size}(R) + \varepsilon m,$$

and explain how large does k need to be. (Big-Oh notation is fine.)