CSE 5095 Research Topics in Big Data Analytics - Spring 2014

Note taker: Tham Hoang

Date: 30th Jan 2014

SELECTION ALGORITHM:

Input: $X = k_1, k_2, \cdots, k_n$

Output: The i^{th} smallest elements of X

Lemma: We can solve this problem in $\tilde{O}(1)$ passes through the data.

Proof:

Chernoff bounds:

Let Y be a binomial random variable (denoted as B(n, p)) with parameters n and p.

Then,

Prob. $[Y > m] \leq \left(\frac{np}{m}\right)^m e^{-np+m} \quad \forall\, m > np$ (1)

Prob. $[Y > (1+\varepsilon)np] \leq \exp(-\varepsilon^2 np/3)$ (2)

Prob. $[Y < (1-\varepsilon)np] \leq \exp(-\varepsilon^2 np/2)$ (3)

A Sampling lemma (Rajasekaran & Reif 1986)

Let X be any set of elements and let S be a random sample of X with $s = |S|$.

Let q be an element of S such that rank(q, S) = j.

Then, the expected rank of q in $X = \frac{n}{s} * j$

Let r_j be the rank of q in X.

Then, Prob. $[|r_j - j * \frac{n}{s}| > \sqrt{3\alpha}\frac{n}{\sqrt{s}}\sqrt{\log n}\,] \leq n^{-\alpha}$.

A Randomized Selection Algorithm

+ Pick a sample S from X. Identity two elements l_1 and l_2 from S whose ranks in S are $\left(i\,\frac{s}{n} - \delta\right)$ and $\left(i\,\frac{s}{n} + \delta\right)$, respectively.

If $\delta \geq \sqrt{4\alpha s \log n}$, then we can show that l_1 and l_2 will bracket the i^{th} smallest element of X with high probability.

Also, $|\{k \in X, l_1 \leq k \leq l_2\}|$ will be 'small' with high probability.

+ Using l_1 and l_2 eliminate all the keys of X that do not have a value in the interval $[l_1, l_2]$. Update the values of n and i.

+ Repeat the above process of sampling and elimination until we are left with $\leq M$ elements. Bring them to memory, do a selection, and output.

To begin with, all keys are alive; /* N is the number of alive keys

Repeat

(1) Add every alive key to the sample with a probability of $\frac{M}{2N}$. Expected number of keys in S = $\frac{M}{2}$. This number is $\leq \frac{3}{4}M$ with high probability.

(2) Pick l_1 and l_2 such that Rank(l_1, S) = $i * \frac{s}{N} - \sqrt{4\alpha s \log n}$ and Rank(l_2, S) = $i * \frac{s}{N} + \sqrt{4\alpha s \log n}$.

(3) Count the number of alive keys that are $< l_1$. Let this be n_1. Count the number of alive keys with a value $\in [l_1, l_2]$. Let this be n_2.

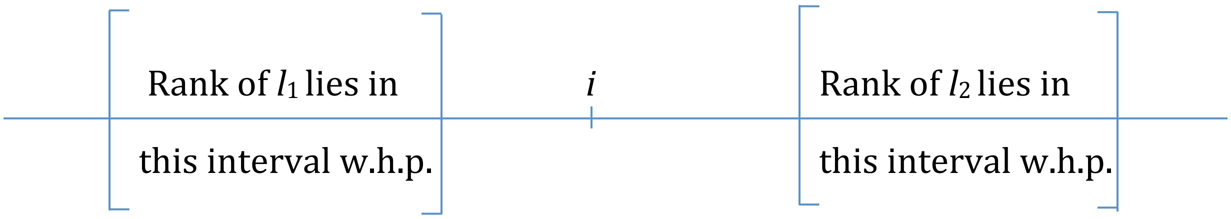
(4) If $i \leq n_1$ or $i > n_1 + n_2$ or $n_2 > \frac{N}{M^{0.4}}$ then go to step 1.

(5) Delete all the alive keys that do not have a value in the range $[l_1, l_2]$.

(6) If $n_2 \leq M$ then quit the repeat loop; $N := n_2$; $i := i - n_1$;

Forever

(7) Find and output the i^{th} smallest from among the alive keys.



$i - \sqrt{4\alpha}\frac{N}{\sqrt{s}}\sqrt{\log N}$ $- \sqrt{3\alpha}\frac{N}{\sqrt{s}}\sqrt{\log N}$

$i - \sqrt{4\alpha}\frac{N}{\sqrt{s}}\sqrt{\log N}$ $+ \sqrt{3\alpha}\frac{N}{\sqrt{s}}\sqrt{\log N}$

$i + \sqrt{4\alpha}\frac{N}{\sqrt{s}}\sqrt{\log N}$ $- \sqrt{3\alpha}\frac{N}{\sqrt{s}}\sqrt{\log N}$

$i + \sqrt{4\alpha}\frac{N}{\sqrt{s}}\sqrt{\log N}$ $+ \sqrt{3\alpha}\frac{N}{\sqrt{s}}\sqrt{\log N}$

If rank(q, S) = j then Prob. $[|r_j - j * \frac{N}{s}| > \sqrt{3\alpha}\frac{N}{\sqrt{s}}\sqrt{\log N}\,] \leq N^{-\alpha}$

⇨ $n_2 \leq 2(\sqrt{4\alpha} + \sqrt{3\alpha})\frac{N}{\sqrt{s}}\sqrt{\log N}$ with high probability, i.e., $n_2 = \tilde{O}\left(\frac{N}{\sqrt{s}}\sqrt{\log N}\right)$.

If N is a polynomial in M then $n_2 = \tilde{O}\left(\frac{N}{M^{0.4}}\right)$.

We will be done in a constant number of repeat loops with high probability.

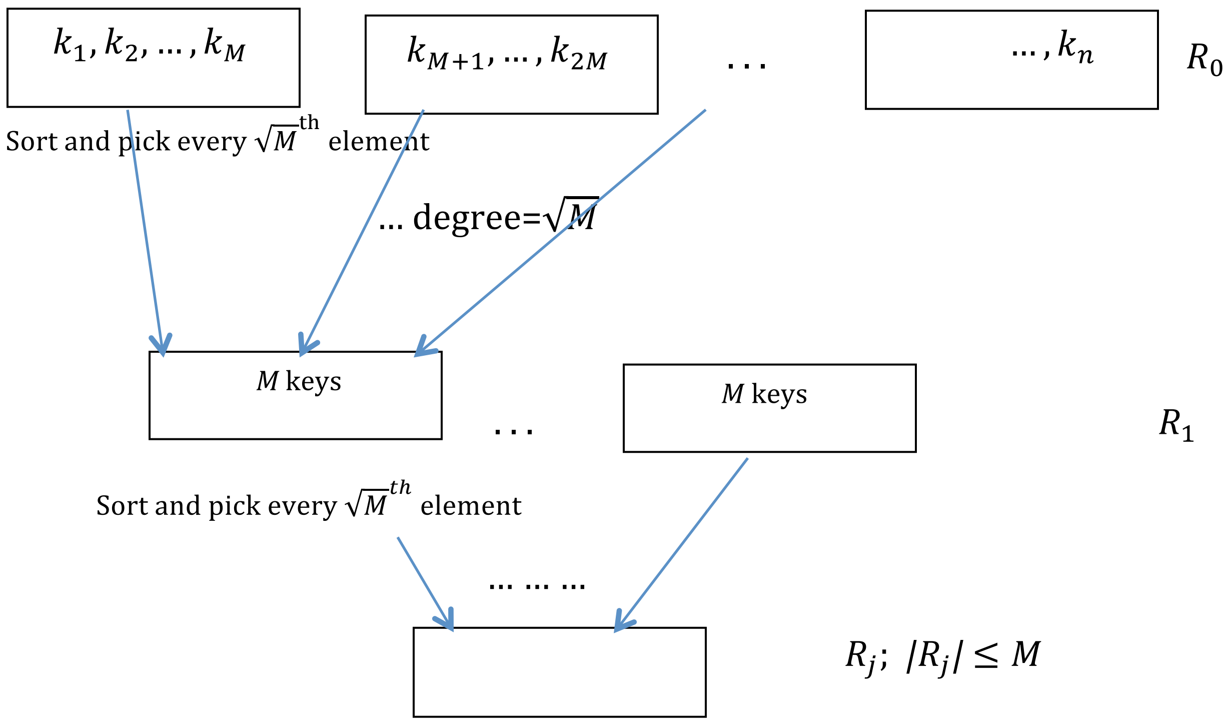
In each iteration of the repeat loop, we spend 2 passes through the ALIVE KEYS.

The number of alive keys decreases by a factor of $M^{0.4}$ in every iteration.

⇨ Total number of passes through the data is ≤ 2 ➔ the number of I/O's is $\leq 2\frac{n}{B}$ with high probability, B being the block size.

A deterministic algorithm (Rajasekaran 2001)

Deterministic sampling



Think of a tree where we have $\frac{n}{M}$ leaves with M keys in each leaf.

Sort each leaf. Each leaf sends its keys with ranks $\sqrt{M}, 2\sqrt{M}, 3\sqrt{M} \dots$ to its parent. Let the leaves be at level 0. We have $\frac{n}{\sqrt{M}}$ keys in level 1. We group these keys into groups of size M each. From each such group $\sqrt{M}$ keys are sent to the next level, and so on.

In general, at each node there are M keys. Each node sorts its keys and sends $\sqrt{M}$ keys to the next level, and so on. Let the root be at level j. $|R_j| \le M$.

We'll pick two keys l_1 and l_2 from R_j such that these two keys will bracket the i^{th} smallest element of X. We eliminate all the keys of X that do have a value in the range $[l_1, l_2]$ and update the values of i and n.

The above process of sampling and elimination continues until the number of remaining keys is no more than *M.* When this happens, we perform an appropriate selection from the remaining keys and output this key.