CSE 5717 Big Data Analytics. Fall 2022

Exam IV Solutions

1. A sampling Lemma stated in class states the following. Let X be any set of n arbitrary real numbers and let S be a random sample of X with s elements. If q is an element of S whose rank in S is j and r_j is the rank of q in X, then:

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

   The above Lemma suggests the following algorithm: 1) Pick a random sample S of X with $|S| = n^{2/3}\log n$; 2) Identify and output the median M of S. This median can be found in $O(|S|)$ time. This implies that the run time of the algorithm is $O(n^{2/3}\log n)$. If r is the rank of M in X, then the above Lemma implies that:

$$Prob\left[\left|r - \frac{n}{2}\right| > \sqrt{3\alpha}n^{2/3}\right] \leq n^{-\alpha}.$$

2. We can employ radix sorting here. Think of each key as a $\log R$-bit binary number. We sort the keys in stages. In each stage we sort them with respect to $\log(M/B)$ bits. Sorting is done starting from the LSBs and moving towards the MSBs.

   In any stage of sorting we have to sort n keys where each key is an integer in the range $\left[1, \frac{M}{B}\right]$. In the core memory we keep $\frac{M}{B}$ buckets, one for each possible value. We bring one block at a time from the disk and distribute the keys in this block into the buckets based on the values of the keys. A key whose value is i will be put into bucket i (for $1 \leq i \leq \frac{M}{B}$. When a bucket gets B keys, this block is written into the disk and the bucket becomes empty. There will be a run in the disk corresponding to every possible value.

   Clearly, there will be $\frac{\log R}{\log(M/B)}$ stages in the algorithm and hence the total number of I/O operations is $O\left(\frac{n}{B}\frac{\log R}{\log(M/B)}\right)$.

3. Construct a suffix tree Q for S in $O(n)$ time. Followed by this, perform an in-order traversal of Q to label every internal node u of Q with an integer $c[u]$ such that $c[u]$ is the number of leaves in the subtree rooted at u.

   Now, perform one more traversal through Q to mark every node whose string depth is $\geq k$. In one additional traversal through Q identify the node u that is marked and whose $c[u]$ is the largest. Finally, output any substring of the path label of u whose length is k.

   Clearly, the total run time of the algorithm is $O(n)$.

4. Note that we generate association rules from frequent itemsets. A frequent itemset occurs in at least one transaction. Let t be any transaction. Since the number of items in t is no more than c, the number of itemsets we can generate out of t is less than 2^c. This implies that the

total number of frequent itemsets is $< n2^c$. Let X be any such frequent itemset. X has at most c items in it. Using the result from Homework 3, problem 4(b), the number of association rules that we can generate from X is $\leq 3^c$. As a result, the total number of association rules we can construct from all possible frequent itemsets is $< n2^c3^c = n6^c = O(n)$.

5. First compute $f_i(x) = (x + a_i)^{2^i}$, for $1 \leq i \leq \log n$. For any i, $(x + a_i)^{2^i}$ can be computed in time $O(2^i)$ (as per Problem 1, Homework 2). Thus all of these polynomials can be computed in a total of $O\left(\sum_{i=1}^{\log n} 2^i\right) = O(n)$ time.

   Followed by this, we do the following:

   $f(x) = f_1(x)$;
   for $i = 2$ **to** $\log n$ **do**
   $f(x) = f(x) \times f_i(x)$;

   Using the theorem that we can multiply two degree d polynomials in $O(d \log d)$ time, the total run time will be $O(\sum_{i=1}^{\log n} 2^i i) = O(n \log n)$.

6. Note that given the activation values for layer l (for $1 \leq l \leq (L-1)$), we can compute the activation values for layer $l+1$ with a matrix-vector multiplication. Specifically, $\vec{a}^{l+1} = \sigma(W^{l+1}\vec{a}^l + \vec{b}^{l+1})$. Here W^{l+1} is the weight matrix (from layer l to layer $l+1$) and is of dimension $n \times n$. $\vec{a}^{l+1}$ and $\vec{a}^l$ are activation vectors and $\vec{b}^{l+1}$ is the bias vector. These are of dimension $n \times 1$ each.

   Consider the computation of $W^{l+1}\vec{a}^l$. Each row of W^{l+1} can be multiplied with $\vec{a}^l$ in $O(\log n)$ time using $\frac{n}{\log n}$ CREW PRAM processors, with the employment of a prefix sums computation. Thus, $W^{l+1}\vec{a}^l + \vec{b}^{l+1}$ can be computed in $O(\log n)$ time using $\frac{n^2}{\log n}$ CREW PRAM processors, given $\vec{a}^l$. Given $W^{l+1}\vec{a}^l + \vec{b}^{l+1}$, we can compute $\vec{a}^{l+1}$ in $O(1)$ time using n processors.

   In summary, one forward propagation step can be completed in $O(L \log n)$ time using $\frac{n^2}{\log n}$ CREW PRAM processors.