CSE 4502/5717 Big Data Analytics. Spring 2019
Exam I Solutions

1. Pick a random element of B and check if this element is in A. Checking can be done using binary search in $O(\log n)$ time. Call these two steps a **phase** of the algorithm. Repeat this phase as many times as it takes to indentify a common element.

   The probability of success in any phase is $\geq \frac{1}{4}$ since we know that there are $\frac{n}{4}$ common elements between A and B. Probability of failure in one phase is $\leq \frac{3}{4}$. Therefore, probability of failing in k successive phases is $\leq \left(\frac{3}{4}\right)^k$. This probability will be $\leq n^{-\alpha}$ if $k \geq \frac{\alpha \log n}{\log(4/3)}$. In other words, the run time of the algorithm is $\widetilde{O}(\log^2 n)$.

2. Assign $\log n$ keys per processor. To begin with the processors attempt to write one of their keys into a memory cell M in parallel. After this write step, every processor reads from M to see which key has been written into. Let x be this key.

   The processors then participate in one more parallel write step where they try write in M a key they have that is not equal to x. As a result, a second distinct element of X is identified. In a similar manner, all the distinct elements are identified. If c is the number of distinct elements, then, all the distinct elements can be identified in $O(\log n)$ time.

   Let the distinct elements be $d_1, d_2, \ldots, d_c$.

   The processors perform a prefix computation to place all the keys equal to d_1 in successive memory cells. (This algorithm was described in class). Followed by this, the processors place all the keys equal to d_2 in successive memory cells; and so on.

   We perform c prefix computations for a total of $O(\log n)$ time.

3. In the main memory keep an array $C[1 : B]$. Also keep buffers $D_1, D_2, \ldots, D_B$, where the size of buffer D_i will be B, for $1 \leq i \leq B$.

   In one pass through the data, count the number of keys in X whose value is i and store this count in $C[i]$, for $1 \leq i \leq B$.

   Perform a prefix sums computation on $C[1], C[2], \ldots, C[B]$ to get $q_1, q_2, \ldots, q_B$. Let $q_0 = 0$.

   In the second pass, bring one block at a time from the disk. Let the keys in the current block be $s_1, s_2, \ldots, s_B$. Distribute these keys to the buffers based on the values of the keys. Specifically, add s_i to D_{s_i}, for $1 \leq i \leq B$. If any of the buffers is full, write it to the disk in the appropriate location and clear the buffer. In particular, the first block whose value is i will be written starting from location $q_{i-1} + 1$, for $1 \leq i \leq B$. Assume that the first block of keys whose value is 1 is written starting from location 1. The second block of keys with a value i will be written statrting from location $q_{i-1} + 1 + B$, and so on.

   When we complete processing the all the input keys in this manner, X would have been written in the disk in sorted order.

The first pass takes $\frac{n}{B}$ read I/O operations (and no write I/O operations). In the second pass also we perform only $\frac{n}{B}$ read I/O operations (and $\sum_{i=1}^{B} \left\lceil \frac{C[i]}{B} \right\rceil \leq \left(\frac{n}{B} + B\right)$ write I/O operations).

4. To solve this problem, we can employ the selection algorithm(s) we have discussed in class. For instance, we showed that we can perform selection on n elements in $O\left(\frac{n}{B}\right)$ I/O operations, B being the block size.

   There will be $\log k$ phases in the algorithm.

   In phase 1, we find the median M of the n elements. We then partition X into X_1 and X_2 using M as the partition element. So far, we have spent $O\left(\frac{n}{B}\right)$ I/O operations. X has been divided into two equal sized parts.

   In phase 2, we find the medians M_1 and M_2 of X_1 and X_2, respectively and partition X_1 into two using M_1 as the partition element. We also partition X_2 into two using M_2 as the partition element. The total number of I/O operations taken will be $O\left(\frac{n}{B}\right)$. X has been divided into 4 equal sized parts.

   We proceed in a similar manner. In phase i we spend $O\left(\frac{n}{B}\right)$ I/O operations, for any $i \geq 1$. At the end of phase i, X will be divided into 2^i equal sized parts. This means that we only need $\log k$ phases. Since we spend $O\left(\frac{n}{B}\right)$ I/O operations in each phase, the total number of I/O operations needed for the entire algorithm is $O\left(\frac{n}{B}\log k\right)$.