CSE 4502/5717 Big Data Analytics. Spring 2026

Exam I Solutions

1. Consider the following algorithm:

   repeat
   Pick a random $j \in [1, n]$;
   if $A[j] < 5$ then output "Type I" and quit;
   if $A[j] > 15$ then output: "Type II" and quit;
   forever;

   Analysis: Consider the case of A being of type I. The probability that $A[j] < 5$ on a radomly picked j is $\frac{1}{3}$. Thus the probability of quitting in any execution of the repeat loop is $\frac{1}{3}$. Therefore, the probability of failure in any execution of the repeat loop is $\frac{2}{3}$. As a result, the probability of failure in the first k iterations of the repeat loop is $\left(\frac{2}{3}\right)^k$. We want this probability to be no more than $n^{-\alpha}$. This happens when $k \geq \frac{\alpha \log n}{\log(3/2)}$. This implies that the run time of this algorithm is $\widetilde{O}(\log n)$, if the array is of type I. A similar analysis holds when the array is of type II.

2. Here is an algorithm:
   repeat
   Flip an n-sided coin to get i;
   Flip an n-sided coin to get j;
   if $i \neq j$ **and** $A[i] = A[j]$ **then output** $A[i]$ and **quit**;
   forever

   Analysis: Call one execution of the **repeat** loop as a basic step. Probability of success in one basic step is $\frac{\frac{n}{2}\left(\frac{n}{10}-1\right)}{n^2} \approx \frac{1}{20}$. This means that the probability of failure in one basic step is $\leq \frac{19}{20}$. Probability of failure in k basic steps is $\leq \left(\frac{19}{20}\right)^k$.

   We want the above probability to be $\leq n^{-\alpha}$. This will happen if $k \geq \frac{\alpha \log n}{\log(20/19)}$. This in turn means that the run time of the above algorithm is $\widetilde{O}(\log n)$.

3. Perform a prefix sums computation on $k_1, k_2, \ldots, k_n$ to get $s_1, s_2, \ldots, s_n$. This can be done in $O(\log n)$ time using $\frac{n}{\log n}$ CREW PRAM processors.

   For the next step, assume that we have n processors. Let $s_0 = 0$.

   for $i = m$ to n **do**
   Processor i computes A_{i-m+1} as $\frac{s_i - s_{i-m}}{m}$.

   The above step takes $O(1)$ time using n CREW PRAM processors. Using the slow-down lemma, this step can be completed in $O(\log n)$ time using $\frac{n}{\log n}$ CREW PRAM processors.

   Thus the entire algorithm takes $O(\log n)$ time using $\frac{n}{\log n}$ CREW PRAM processors.

4. 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 to 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.

5. Keep a priority queue Q with B keys (with an initial value of ∞) in the main memory.

   repeat
   Bring the next block U from the disk;
   for each key k in U **do**
   Let m be the maximum key in Q;
   if $k < m$ **then** delete m from Q and insert k;
   until all the input keys have been processed
   Output all the B keys in Q

6. We can sort X to get X' and write X' in the disk. This will take $O\left(\frac{n}{B}\frac{\log(n/M)}{\log(M/B)}\right)$ I/O operations as was shown in class. Likewise, sort Y to get Y' and write Y' in the disk. This can also be done in $O\left(\frac{n}{B}\frac{\log(n/M)}{\log(M/B)}\right)$ I/O operations.

   (a) Now bring the first block A of X' and the first block C of Y' from the disk.

   (b) Compare A and C. If they have a common element, report the common element and quit. If they don't have a common element, let the last element of A be a and the last element of C be c.

   (c) If $a > c$ then bring the next block C' from Y' and let $C = C'$. If $a < c$ then bring the next block A' from X' and let $A = A'$.

   (d) Repeat steps (b) and (c) until all the elements of either X' or Y' or both have been brought into the main memory and processed.

   (e) Report that X and Y do not have a common element.