CSE 4502/5717 Big Data Analytics. Fall 2019

Exam I Solutions

1. 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 $\widetilde{O}(\log n)$.

2. Pick a random sample S of size $c\alpha \log n$ from A. If S has only zeros, output **Type I**. If S has only ones, output **Type II**. Otherwise output **Type III**.

   If the input is of type I or II, note that the above algorithm will never output an incorrect answer. However, if the input is of type III, the algorithm might give an incorrect answer. An incorrect answer will be given if S has either only zeros or only ones.

   Assume that the input is of type III. Let P_0 be the probability that S has only zeros and let P_1 be the probability that S has all ones. Clearly, $P_0 \leq \left(\frac{3}{4}\right)^{c\alpha \log n}$. P_0 will be $\leq n^{-2\alpha}$ if $c \geq \frac{2}{\log(4/3)}$. Also, $P_1 \leq \left(\frac{1}{4}\right)^{c\alpha \log n}$. P_1 will be $\leq n^{-2\alpha}$ if $c \geq 1$.

   The probability of an incorrect answer is $\leq P_1 + P_2$. This will be $\leq 2n^{-2\alpha} \leq n^{-\alpha}$ if $c \geq \frac{2}{\log(4/3)}$.

3. Let $X = k_1, k_2, \ldots, k_n$ be the input sequence. Here is an algorithm that checks if X has been sorted in nondecreasing order:

   Processor 1 sets *Result* to 1;

   for $i = 1$ **to** $n - 1$ **in parallel do**

   if $k_i > k_{i+1}$ **then** processor i tries to write 0 in *Result*;

   The correctness of the algorithm is clear. Also, the algorithm runs in $O(1)$ time using $n - 1$ CRCW PRAM processors.

4. Assume that we have $2n$ processors. Let $X = a_1, a_2, \ldots, a_n$ and $Y = b_1, b_2, \ldots, b_n$. Each of the input elements is assigned one processor. Let the processors be labelled $p_1, p_2, \ldots, p_{2n}$. The following algorithm merges X and Y:

   for each input key k **in parallel do**

   if $k = a_i$ for some i, then the processor associated with k performs a binary search in Y to figure out the number q of elements in Y that are less than k;

The global rank of k is computed as $r_k = (i-1) + q + 1 = i + q$;

if $k = b_j$ for some j, then the processor associated with k performs a binary search in X to figure out the number q of elements in X that are less than k;

The global rank of k is computed as $r_k = (j-1) + q + 1 = j + q$;

Write k in memory cell r_k;

Analysis: The binary search in the above algorithm takes $O(\log n)$ time. The other operations take a total of $O(1)$ time. Thus the run time of the entire algorithm is $O(\log n)$.

5. 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.

6. Use the selection algorithm discussed in class to identify the element k_1 of X whose rank in X is $\frac{n}{2} - \frac{M}{2}$. This will take $O\left(\frac{n}{B}\right)$ I/O operations. Likewise identify the element k_2 of X whose rank in X is $\frac{n}{2} + \frac{M}{2}$. This will take $O\left(\frac{n}{B}\right)$ I/O operations as well.

   Now do one more pass through the data to collect all the keys of X that have a value in the range $[k_1, k_2]$. This can be done by bringing one block of X at a time into the main memory.

   The I/O complexity of the entire algorithm is thus $O\left(\frac{n}{B}\right)$.