CSE 4502/5717 Big Data Analytics. Fall 2022

Exam I Solutions

1. Consider the following algorithm:

```
repeat
  for $i = 1$ to $\sqrt{n}$ do
      Flip a $\sqrt{n}$-sided coin to get $u$;
      Flip a $\sqrt{n}$-sided coin to get $v$;
      if $u \neq v$ and $A_i[u] = A_i[v]$ then output $A_i[u]$ and quit;
forever
```

   Analysis: Note that since $\sum_{i=1}^{\sqrt{n}} n_i = n^{0.8}$, there should exist a k ($1 \leq k \leq \sqrt{n}$) such that $n_k \geq n^{0.3}$. The **for** loop is executed for every array. Call one iteration of the **for** loop as a basic step.

   Focus now only on A_k. Probability of successfully identifying the repeated element in A_k in one basic step $= \frac{n^{0.3}(n^{0.3}-1)}{\sqrt{n}\times\sqrt{n}} \approx \frac{1}{n^{0.4}}$. Probability of failure in one basic step is $\approx 1 - \frac{1}{n^{0.4}}$. Probability of failure in q successive steps $\approx \left(1 - \frac{1}{n^{0.4}}\right)^q$. We want this to be $\leq n^{-\alpha}$. This happens when $q \geq \alpha n^{0.4} \log_e n$.

   Since each basic step takes $O(\sqrt{n})$ time, the run time of the entire algorithm is $\widetilde{O}(n^{0.9} \log n)$.

2. Let $X = k_1, k_2, \ldots, k_n$ be the input. We can find the median M of X in $\Theta(n)$ time using the BFPRT algorithm. Followed by this, we partition X into X_1 and X_2 where $X_1 = \{q \in X : q < M\}$ and $X_2 = \{q \in X : q > M\}$. This partitioning takes $\Theta(n)$ time. Subsequently, we recursively sort X_1 and X_2. Finally, we output sorted(X_1), M, sorted(X_2).

   Note that $|X_1| = |X_2| = n/2$ (since M is the median of X). Let $T(n)$ be the run time of the above algorithm on any input of size n. Then, the recurrence relation for $T(n)$ will be:

$$T(n) = 2T(n/2) + \Theta(n).$$

   Using the Master theorem, we can solve the above recurrence relation to get: $T(n) = \Theta(n \log n)$.

3. We use an array $M[1 : n^{2/3}]$. Consider the following algorithm:

```
1) for $i = 1$ to $n^{2/3}$ in paralel do
     Processor $i$ writes $-\infty$ in $M[i]$;
2) for $i = 1$ to $n$ in paralel do
     Processor $i$ writes $k_i$ in $M[k_i]$;
3) All the $n$ processors collectively find and output the maximum of
     $M[1], M[2], \ldots, M[n^{2/3}]$.
```

Analysis: Steps 1 and 2 take one unit of time each. In step 3 we are finding the maximum of $n^{2/3}$ elements using n common CRCW PRAM processors and hence this can be done in $O(1)$ time. (Note that we proved in class that we can find the maximum of N elements in $O(1)$ time using $N^{1.5}$ common CRCW PRAM processors). Thus, the total run time of this algorithm is $O(1)$ using n common CRCW PRAM processors.

4. 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 $s_i - s_{i-m}$.
   ```

   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.

   Now, perform a prefix minima computation on $A_1, A_2, \ldots, A_{n-m+1}$ to get $B_1, B_2, \ldots, B_{n-m+1}$. Output B_{n-m+1}. This step can be done 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.

5. We can use the k-way merge algorithm that we have discussed in class, with $k = M/B$. We utilize a complete k-ary tree with q leaves. We start from the leaves and move up the tree. At each internal node, we merge the k runs coming from the children. When the root merges the runs from the children, we get the final output.

   Merging at each internal node can be done in exactly the same manner that we discussed in class. As a result, at each level of the tree we do $O(n/B)$ I/O operations and there are $\frac{\log q}{\log(M/B)}$ levels. Put together, the total I/O complexity of the algorithm is $O\left(\frac{n}{B}\frac{\log q}{\log(M/B)}\right)$.

6. Perform selection on X three times to find elements of X whose ranks in X are $\frac{n}{4}$, $\frac{n}{2}$, and $\frac{3n}{4}$. Let these elements be q_1, q_2, and q_3, respectively. These selections can be done in a total of $O\left(\frac{n}{B}\right)$ I/O operations using the out-of-core version of the BFPRT algorithm.

   Now perform one more pass through the data to generate $X_1 = \{q \in X : q \leq q_1\}$, $X_2 = \{q \in X : q_1 < q \leq q_2\}$, $X_3 = \{q \in X : q_2 < q \leq q_3\}$, and $X_4 = \{q \in X : q > q_3\}$. This can be done by keeping four output buffers in the core memory (one each for X_1, X_2, X_3, and X_4, respectively). Bring one block from X; Distribute the elements of this block into the four output buffers based on the values of these elements. If any of the output buffers is full, output this buffer into the corresponding sequence in the disk. Repeat this process until all the elements of X have been acted on. Clearly, this takes only one pass.

   Overall, the I/O complexity of this algorithm is $O\left(\frac{n}{B}\right)$.