CSE 5500 Algorithms. Fall 2018

Solutions to Exam 1.

1. Consider the following algorithm:
   for $i := 1$ **to** $\alpha\sqrt{n}\log_e n$ **do**

   Pick a random $j \in [1, n]$ and pick a random $k \in [1, n]$; If $j \neq k$ and $a[j] = a[k]$ then output: "Type II" and quit;

   Output: "Type I";

   Analysis: Note that if the array is of type I, the above algorithm will never give an incorrect answer. Thus assume that the array is of type II. We'll calculate the probability of an incorrect answer as follows.

   Probability of coming up with the correct answer in one iteration of the for loop is $\frac{n^{3/4}(n^{3/4}-1)}{n^2} \approx \frac{1}{\sqrt{n}}$. Thus, probability of failure in any iteration is $1 - \frac{1}{\sqrt{n}}$. As a consequence, probability of failure in q successive iterations is $\left(1 - \frac{1}{\sqrt{n}}\right)^q \leq \exp(-q/\sqrt{n})$ (using the fact that $(1-1/x)^x \leq 1/e$ for any $x > 0$). This probability will be $\leq n^{-\alpha}$ when $q \geq \alpha\sqrt{n}\log_e n$.

   Thus the output of this algorithm is correct with a high probability.

2. Here we utilize the linear time selection algorithm. Find the median M of X in $O(n)$ time. Partition X into two parts X' and X'' where all the elements in X_1 are $\leq M$ and the elements of X_2 are $> M$. Now recursively partition X' into $\frac{k}{2}$ equal ordered parts and X'' also into $\frac{k}{2}$ equal ordered parts. At each level of recursion we spend $O(n)$ time and there are $\log k$ levels. Therefore, the total run time is $O(n \log k)$.

3. We can make use of random sampling here. The idea is to pick a random sample S of size $s = \sqrt{n}$ from X, find the median M of S, and output this median. Clearly, the run time of the algorithm is $O(\sqrt{n})$. What can we say about the rank of M in X? Using the sampling lemma stated in class, the rank of M lies in the interval $\left[\frac{n}{2} - \sqrt{4\alpha}\frac{n}{\sqrt{s}}\sqrt{\log n}, \frac{n}{2} + \sqrt{4\alpha}\frac{n}{\sqrt{s}}\sqrt{\log n}\right]$ with probability $\geq (1 - n^{-\alpha})$. I.e., the rank of M in X lies in the interval $\left[\frac{n}{2} - \sqrt{4\alpha}\ n^{3/4}\sqrt{\log n}, \frac{n}{2} + \sqrt{4\alpha}\ n^{3/4}\sqrt{\log n}\right]$ with probability $\geq (1 - n^{-\alpha})$.

4. Without loss of generality assume that $k = 2^m$ for some integer m. Consider a full binary tree with k leaves where leaf i has X_i, for $1 \leq i \leq k$. Let the root of the tree be at level 0. Then, the leaves are level m. We merge all the k sequences using the following algorithm. (Before merging the sequences we replace each key k^i_j from X_i with (k^i_j, i), for $1 \leq i \leq k$ and $1 \leq j \leq n$):

   for $i = (m-1)$ **down to** 0 **do**
   for each node N of level i **do**
   Merge the sequences of the two children of N;

   When the above algorithm completes we get a sorted sequence Y which is a merge of all the k input sequences (where we keep track of the sequence that each key comes from). Scan through the sorted sequence to check if there is an element common to all the sequences.

   Merging takes $O(kn)$ time per level and there are $\log k$ levels. Thus the total merge time is $O(nk \log k)$. Scanning takes $O(nk)$ time and hence the total run time is $O(nk \log k)$.

5. Sort each of the sets $S_1, S_2, \ldots, S_{\log n}$ independently using any asymptotically optimal general sorting algorithm (such as heapsort). The time needed to sort each of these sets is $O\left(\frac{n}{\log^2 n}\log\left(\frac{n}{\log^2 n}\right)\right) = O\left(\frac{n}{\log n}\right)$. Thus we can sort each of the sets $S_1, S_2, \ldots, S_{\log n}$ in a total of $O(n)$ time.

   Replace each element in each of the sets $S_{1+\log n}, S_{2+\log n}, \ldots, S_k$ with a tuple as follows. If $S_i = \{k^i_1, k^i_2, \ldots, k^i_{n_i}\}$ where $n_i = |S_i|$, then generate a sequence $X_i = (i, k^i_1), (i, k^i_2), \ldots, (i, k^i_{n_i})$, for $(1 + \log n) \leq i \leq k$. Now sort the sequence $X_{1+\log n}, X_{2+\log n}, \ldots, X_k$. Since there are $n - \frac{n}{\log n}$ elements in X and the elements of X are integers in the range $[1, n^{23}]$, this sorting can be done in $O(n)$ time.

From the above sorted sequence we can obtain each of the input sets $S_{1+\log n}, S_{2+\log n}, \ldots, S_k$ in sorted order.

The total run time is $O(n)$.

6. Let $\mathcal{S}$ be a subset of the field $\mathcal{F}$. We pick a random element r of $\mathcal{S}$ and check if $(f(r))^n = (g(r)))^m$. If yes, we output "Yes" else we output "No".

   We can evaluate $f(r)$ and $g(r)$ in times $O(m)$ and $O(n)$, respectively. From $f(r)$ we can compute $(f(r))^n$ in $O(\log n)$ time. Also, given $g(r)$, we can compute $(g(r))^m$ in $O(\log m)$ time. Thus the total run time of the algorithm is $O(m+n)$.

   The degree of both the polynomials $(f(x))^n$ and $(g(x))^m$ is mn. As per one of the theorems we proved in class, the probability that the above algorithm gives an incorrect answer is $\leq \frac{mn}{|\mathcal{S}|}$. This probability will be $\leq n^{-\alpha}$ if $|\mathcal{S}| \geq mn^{\alpha+1}$.