CSE 5500 Algorithms. Fall 2018

Exam I (Model) Solutions

1. Here is a Las Vegas algorithm:

   repeat

   1) Pick a random sample S from $A[1:n]$ of size $8\alpha\sqrt{n}\log n$;

   2) Sort the sample S, scan through it to see if there are multiple copies of any element. If so, output this element and quit;

   forever

   Analysis: Let X be the number of copies of the repeated element in the sample. Clearly, X is binomially distributed with parameters $8\alpha\sqrt{n}\log n$ and $\frac{\sqrt{n}}{n}$. The mean of X is $8\alpha\log n$. Using Chernoff bounds, $Pr[X < (1-\epsilon)8\alpha\log n] \leq \exp\left(-\frac{\epsilon^2 8\alpha\log n}{2}\right)$. Picking $\epsilon = 1/2$, $Pr[X < 4\alpha\log n] \leq n^{-\alpha}$. Thus it follows that the **repeat** loop is executed only once with high probability.

   Step 1 takes $O(\sqrt{n}\log n)$ time. Sorting in step 2 takes $O(\sqrt{n}\log^2 n)$ time. Looking for the repeated element in the sorted sample takes $O(\sqrt{n}\log n)$ time. Put together, the run time of the algorithm is $\widetilde{O}(\sqrt{n}\log^2 n)$ time.

2. Here is a Monte Carlo algorithm: Pick a random sample S of $k\alpha\log n$ elements from X, find and output the element of S whose rank in S is $\frac{3s}{8}$, where $s = |S|$. We can use the BFPRT algorithm to select this element of S. Clearly, this algorithm runs in $O(\log n)$ time. We now have to show that this output will be correct with high probability.

   Let M be the output of the algorithm. We know that $\text{rank}(M, S) = \frac{3s}{8}$. Let r_M be the rank of M in X. Using the sampling lemma (in item 6 of the Helpsheet), $\text{Prob.}\left[|r_M - \frac{3s}{8}\frac{n}{s}| > \sqrt{3\alpha}\frac{n}{\sqrt{s}}\sqrt{\log n}\right] \leq n^{-\alpha}$. For a choice of $k = 192$, this inequality becomes: $\text{Prob.}\left[|r_M - \frac{3n}{8}| > \frac{n}{8}\right] \leq n^{-\alpha}$. $\square$

3. Let the run time of $\mathcal{A}$ be $T_\mathcal{A}$ and that of $\mathcal{B}$ be $T_\mathcal{B}$. Then the recurrence relation for $T_\mathcal{A}$ is: $T_\mathcal{A}(n) = 36T_\mathcal{A}(n/6) + \Theta(n)$. Here $a = 36, b = 6$, and $f(n) = \Theta(n)$. $n^{\log_b a} = n^2$. Case 1 of Master theorem applies. Thus, $T_\mathcal{A}(n) = \Theta(n^2)$.

   The recurrence relation for $T_\mathcal{B}$ is: $T_\mathcal{B}(n) = \sqrt{n}T_\mathcal{B}(\sqrt{n}) + n$. Master theorem does not apply. We can use repeated substitutions here. Assume that the base case is: $T_\mathcal{B}(n) = 1$ for $n \leq 4$. We have:

$$T_\mathcal{B}(n) = \sqrt{n}T_\mathcal{B}(\sqrt{n}) + n = \sqrt{n}[n^{1/4}T_\mathcal{B}(n^{1/4}) + \sqrt{n}] + n = n^{1-1/2^2}T_\mathcal{B}(n^{1/2^2}) + 2n$$

$$= n^{1-1/2^3}T_\mathcal{B}(n^{1/2^3}) + 3n.$$

After making $i-1$ substitutions we see that: $T_{\mathcal{B}}(n) = n^{1-1/2^i} T_{\mathcal{B}}(n^{1/2^i}) + in$. The base case is reached when $n^{1/2^i} = 4$, i.e., when $i = \log\log n - 1$. Substituting this value of i in the general expression we see that $T_{\mathcal{B}}(n) = \Theta(n \log\log n)$.

As a result, $\mathcal{B}$ has a better run time than $\mathcal{A}$ and hence $\mathcal{B}$ is preferable.

4. We can first scan through the elements of X and partition them into four parts X_a, X_b, X_c, and X_d, where $X_a = \{q : q \in X \text{ and } q \in [a, a + n^{10}]\}$, $X_b = \{q : q \in X \text{ and } q \in [b, b + n^{20}]\}$, and so on. This partitioning can be done with at most 6 comparisons per input key and hence this partitioning takes a total of $O(n)$ comparisons. After partitioning X we sort X_a, X_b, X_c, and X_d separately and independently. To sort X_a, we subtract a from each key of X_a. Let the resultant sequence be X'_a. Clearly, the keys in X'_a are integers in the range $[0, n^{10}]$ and hence can be sorted in $O(|X_a|)$ time using the integer sorting algorithm. In a similar manner we sort X_b, X_c, and X_d in time $O(|X_b|), O(|X_c|)$, and $O(|X_d|)$, respectively. As a result, the total time needed to sort these four parts of X is $O(n)$.

   Finally, we output X_a in sorted order followed by X_b in sorted order, X_c in sorted order, and X_d in sorted order. The total run time of the entire algorithm is $O(n)$.

5. Let X be the given sequence of n keys. We partition X into groups $G_1, G_2, \ldots, G_{n/3}$ of size 3 each. Let the medians of these groups be $M_1, M_2, \ldots, M_{n/3}$, respectively. Let M be the median of these medians. We then employ the quickselect algorithm with M as the pivot. Let $X_1 = \{q \in X : q < M\}$ and $X_2 = \{q \in X : q > M\}$. There are $n/6$ groups for which the medians are $\leq M$. In each such group there will be at least 2 elements that are $\leq M$. Therefore, at least $n/3$ elements of X will be $\leq M$. This means that $|X_2| \leq \frac{2}{3}n$ and for similar reasons $|X_2| \leq \frac{2}{3}n$.

   Thus, if $T(n)$ is the run time of this algorithm on any input of size n and for any i, we have: $T(n) \leq T\left(\frac{n}{3}\right) + T\left(\frac{2}{3}n\right) + \Theta(n)$. By induction we see that $T(n) = O(n \log n)$.

6. Let $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$. Construct two polynomials $f(x) = \Pi_{i=1}^n (x - a_i)$ and $g(x) = \Pi_{i=1}^n (x - b_i)$. The problem of checking if A and B are identical can be reduced to the problem of checking if $f(x)$ and $g(x)$ are identical. We can use fingerprinting to do this in $O(n)$ time as follows. Let $\mathcal{S}$ be the set of integers in the range $[1, n^{\alpha+1}]$. Pick a random integer r from $\mathcal{S}$, evaluate $f(r)$ and $g(r)$, and check if $f(r) = g(r)$. If $f(r) = g(r)$, then output: "A and B are identical"; else output: "A and B are not identical".

   Clearly, if $f(r) \neq g(r)$, then A and B are not identical. If A and B are identical, then the above algorithm will never give an incorrect answer. If A and B are not identical, what is the probability that $f(r) = g(r)$? Note that the polynomial $h(x) = f(x) - g(x)$ has at most n distinct zeros. Therefore, $Prob.[f(r) = g(r)] \leq \frac{n}{n^{\alpha+1}} = n^{-\alpha}$.