Name: ________________________________

CSE 5500 Algorithms

Exam I; October 16, 2018

Note: You are supposed to give proofs to the time bounds of your algorithms. Read the questions carefully before attempting to solve them.

1. (16 points) Input is an array $a[1:n]$ of arbitrary real numbers. The array could only be of one of the following two types: 1) **Type I:** All the elements in the array are distinct; or 2) **Type II:** The array has $n^{3/4}$ copies of one element, the other elements being distinct. Present a Monte Carlo algorithm that determines the type of the array in $O(\sqrt{n}\log n)$ time. Show that the output of your algorithm will be correct with a high probability.

2. (17 points) Input are a sequence X of n arbitrary and distinct real numbers and an integer $k < n$. The problem is to partition X into $X_1, X_2, \ldots, X_k$ such that $|X_i| = \frac{n}{k}$ for $1 \leq i \leq k$ and all the elements of X_j are less than any element in X_{j+1}, for $1 \leq j \leq k-1$. Present an $O(n \log k)$ time algorithm to solve this problem.

3. (17 points) Input is a sequence X of n arbitrary and distinct real numbers. The problem is to identify an *approximate median* of X. An element $x \in X$ is said to be an approximate median of X if the rank of x in X lies in the interval $\left[\frac{n}{2} - cn^{3/4}\log n, \frac{n}{2} + cn^{3/4}\log n\right]$ for some constant $c > 0$. Present an $O(\sqrt{n})$-time Monte Carlo algorithm to find an approximate median of X. Prove that the output of your algorithm will be correct with high probability.

4. (17 points) Input are sorted sequences $X_1, X_2, \ldots, X_k$ each of length n. The problem is to check if there is an element common to all the k sequences. Present an algorithm to solve this problem in $O(nk \log k)$ time.

5. (17 points) Input are k sets $S_1, S_2, \ldots, S_k$ such that $\sum_{i=1}^{k} |S_i| = n$. The sets $S_1, S_2, \ldots, S_{\log n}$ have $\frac{n}{\log^2 n}$ arbitrary real numbers each. The elements of the sets $S_{1+\log n}, S_{2+\log n}, \ldots, S_k$ are integers in the range $[1, n^{23}]$. The problem is to sort these k sets. Present an algorithm to sort all of these sets in a total of $O(n)$ time.

6. (16 points) Input are two polynomials $f(x)$ and $g(x)$ of degree m and n, respectively. The problem is to check if $(f(x))^n = (g(x))^m$. Present a Monte Carlo algorithm to solve this problem in $O(m+n)$ time. Show that the output of your algorithm will be correct with a high probability.

CSE 5500 Algorithms
Fall 2018; Exam I – Helpsheet

1. **Preliminaries.** We say $f(n) = O(g(n))$ if $f(n) \leq cg(n)$ for all $n \geq n_0$ for some constants c and n_0. We say $f(n) = \Omega(g(n))$ if and only if $g(n) = O(f(n))$. Also, $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

   A partial list of functions in increasing order is: $O(1)$, $(\log n)^\epsilon$, $\log n$, $(\log n)^{1+\mu}$, n^ϵ, n, $n^{1+\mu}$, 2^{n^ϵ}, 2^n, $2^{n^{1+\mu}}$ where $0 < \epsilon < 1$ and $\mu > 0$ are constants.

   Stirling's approximation: $n! \approx (n/e)^n \sqrt{2\pi n}$.

   $\sum_{i=1}^n i = n(n+1)/2$. $\sum_{i=1}^n i^2 = n(n+1)(2n+1)/6$. $\sum_{i=1}^n i^3 = n^2(n+1)^2/4$.

2. **Master theorem.** Consider the recurrence relation: $T(n) = aT(n/b) + f(n)$, where $a \geq 1$ and $b > 1$ are constants. **Case 1:** If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$. **Case 2:** If $n^{\log_b a} = \Theta(f(n))$, then $T(n) = \Theta(f(n) \log n)$. **Case 3:** If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and $af(n/b) \leq cf(n)$ for some constant $c < 1$, then, $T(n) = \Theta(f(n))$.

3. **Randomized algorithms.** A Monte Carlo algorithm runs for a prespecified amount of time and its output is correct with high probability. By high probability we mean a probability of $\geq (1 - n^{-\alpha})$, for any constant α. A Las Vegas algorithm always outputs the correct answer and its run time is a random variable. We say the run time of a Las Vegas algorithm is $\widetilde{O}(f(n))$ if the run time is $\leq c\alpha f(n)$ for all $n \geq n_0$ with probability $\geq (1 - n^{-\alpha})$, for some constants c and n_0.

   Chernoff bounds: If X is a random variable with a binomial distribution $B(n, p)$, then $Pr[X \geq (1+\epsilon)np] \leq \exp(-\epsilon^2 np/3)$ and $Pr[X \leq (1-\epsilon)np] \leq \exp(-\epsilon^2 np/2)$, for any $0 < \epsilon < 1$.

4. **Sorting.** The mergesort algorithm has a worst case run time of $O(n \log n)$. The run time of quicksort is $O(n^2)$ in the worst case and $O(n \log n)$ on the average.

   Frazer-McKellar's randomized algorithm does sorting using $n \log n + \widetilde{O}(n \log \log n)$ comparisons.

   Radix or Integer Sorting: We can sort n keys in $O(n)$ time if the keys are integers in the range $[1, n^c]$ (for any constant c).

5. **Selection:** This problem takes as input a sequence $X = k_1, k_2, \ldots, k_n$ of arbitrary real numbers and an integer $i, 1 \leq i \leq n$. The problem is to identify the ith smallest element of X. The quickselect algorithm runs in $O(n^2)$ time in the worst case and $O(n)$ time on the average. BFPRT algorithm runs in $O(n)$ time in the worst case. The randomized algorithm of Floyd and Rivest makes only $n + \min\{i, n-i\} + \widetilde{o}(n)$ comparisons. In this context we used the following lemma: Let X be any set of arbitrary real numbers and let S be a random sample of X with $|S| = s$. Let q be an emelent of S such that $\text{rank}(q, S) = j$. If r_j is the rank of q in X, then the following holds: $\text{Prob.}\left[|r_j - j\frac{n}{s}| > \sqrt{4\alpha}\frac{n}{\sqrt{s}}\sqrt{\log n}\right] \leq n^{-\alpha}$, for any $\alpha > 0$.

6. **Fingerprinting:** The technique of fingerprinting can be applied to check if two given objects are the same. Instead of comparing the objects directly, we apply a function on each and compare the images. Some examples we have seen are: 1) For three given $n \times n$ matrices A, B, and C, the problem is to check if $AB = C$. We pick a random vector r from $\{0,1\}^n$ and check if $A(Br) = Cr$. If so, we output: "$AB = C$"; else we output "$AB \neq C$". If $AB \neq C$, we showed that $Prob.[A(Br) = Cr] \leq 1/2$; 2) Given three polynomials $f(x), g(x)$, and $h(x)$, the problem is to check if $h(x) = f(x) \times g(x)$. The idea of fingerprinting can be used as follows: we pick a random element r from $\mathcal{S}$ (which is a subset of the field $\mathcal{F}$) and check if $f(r) \times g(r) = h(r)$. If so, we output: "$h(x) = f(x) \times g(x)$"; else we output: "$h(x) \neq f(x) \times g(x)$". We can show that the probability of an incorrect answer is no more than $\frac{d}{|S|}$ where d is the degree of $h(x) - f(x) \times g(x)$; 3) Schwartz-Zippel theorem extends the above technique to check if a given multivariate polynomial is identically zero; 4) Edmonds' theorem in conjunction with Schwartz-Zippel theorem can be used to check for the existence of perfect matchings in graphs; 5) Karp-Rabin's algorithm applies fingerprinting to the string matching problem. To check if two given n-bit integers A and B are the same, the basic idea of this algorithm is to pick a random prime $p \leq t$ and check if $A \bmod p = B \bmod p$. Probability of an incorrect answer is $O\left(\frac{n}{t/(\log t)}\right)$.