CSE 4502/5717 Big Data Analytics; Fall 2024
Exam 1 Helpsheet

1. **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 (n being the input size). 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.

   For the repeated element identification problem we devised a Las Vegas algorithm with a run time of $\widetilde{O}(\log n)$. Given a sequence of n elements, we presented a Monte Carlo algorithm to find an element $\geq$ the median that runs in time $O(\log n)$. We also showed that sorting can be done in $n \log n + \widetilde{O}(n \log \log n)$ comparisons (using the idea of Frazer and McKellar).

2. **Master theorem.** Consider the recurrence relation: $T(n) = aT(n/b) + f(n)$, where $a \geq 1$ and $b > 1$ are constants. **Case1:** If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$. **Case2:** If $n^{\log_b a} = \Theta(f(n))$, then $T(n) = \Theta(f(n) \log n)$. **Case3:** 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. **PARALLEL ALGORITHMS**. The model we used was the PRAM (Parallel Random Access Machine). Processors communicate by writing into and reading from memory cells that are accessible to all. Depending on how read and write conflicts are resolved, there are variants of the PRAM. In an Exclusive Read Exclusive Write (EREW) PRAM, no concurrent reads or concurrent writes are permitted. In a Concurrent Read Exclusive Write (CREW) PRAM, concurrent reads are permitted but concurrent writes are prohibited. In a Concurrent Read Concurrent Write (CRCW) PRAM both concurrent reads and concurrent writes are allowed. Concurrent writes can be resolved in many ways. In a Common CRCW PRAM, concurrent writes are allowed only if the conflicting processors have the same message to write (into the same cell at the same time). In an Arbitrary CRCW PRAM, an arbitrary processor gets to write in cases of conflicts. In a Priority CRCW PRAM, write conflicts are resolved on the basis of priorities (assigned to the processors at the beginning).

   We presented a Common CRCW PRAM algorithm for finding the Boolean AND of n given bits in $O(1)$ time. We used n processors. As a corollary we gave an algorithm for finding the minimum (or maximum) of n given arbitrary real numbers in $O(1)$ time using n^2 Common CRCW PRAM processors.

   We also discussed an optimal CREW PRAM algorithm for the prefix computation problem. This algorithm uses $\frac{n}{\log n}$ processors and runs in $O(\log n)$ time on any input of n elements. (For the prefix computation problem the input is a sequence of elements from some domain Σ: $k_1, k_2, \ldots, k_n$ and the output is another sequence: $k_1, k_1 \oplus k_2, \ldots, k_1 \oplus k_2 \oplus k_3 \oplus \cdots \oplus k_n$, where $\oplus$ is any binary associative and unit-time computable operation on Σ.) As an application of prefix computation, we proved that sorting of n elements can be done in $O(\log n)$ time using $\frac{n^2}{\log n}$ CREW PRAM processors.

   The slow-down Lemma: If $\mathcal{A}$ is a parallel algorithm that uses P PRAM processors and runs in T time, then $\mathcal{A}$ can be run on a P'-processor machine to get a run time of T' such that $T' = O\left(\frac{PT}{P'}\right)$, for any $P' \leq P$.

4. In an out-of-core computing model we typically measure only the number of I/O operations (i.e., the I/O complexity) performed by any algorithm. Computing time normally is much less than the I/O time. We let M and B denote the size of the core memory and the block size, respectively. We showed the following results for a single disk model: 1) We can sort N elements with $O\left(\frac{N}{B}\frac{\log(N/M)}{\log(M/B)}\right)$ I/O operations. We first formed runs of length M each and then merged these N/M runs using a M/B-way merge algorithm; 2) There exists a deterministic algorithm for selection whose I/O complexity is $O(N/B)$. BFPRT algorithm was used to achieve this result.