CSE 5500 Algorithms
Fall 2018; Exam III; Help Sheet

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 concuurent 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. In addition, we proved that the maximum of n arbitrary elements can be found in $O(\log\log n)$ time using n common CRCW PRAM processors and that the maximum of n integers in the range $[1, n^c]$ can be found in $O(1)$ time using n common CRCW PRAM processors, c being any constant.

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. We showed that we can sort n elements in $\widetilde{O}(\log n)$ time using n arbitrary CRCW PRAM processors.

INTRACTABLE PROBLEMS. A problem π_1 is said to be polynomially reducible to another problem π_2 (denoted as $\pi_1 \propto \pi_2$) if the following statement holds: "If π_2 can be solved in deterministic polynomial time then π_1 can also be solved in deterministic ploynomial time".

A problem π is said to be $\mathcal{NP}$-hard if $\pi' \propto \pi$ for every $\pi' \in \mathcal{NP}$. Equivalently, a problem π is $\mathcal{NP}$-hard if $\pi' \propto \pi$ where π' is known to be $\mathcal{NP}$-hard. A problem π is $\mathcal{NP}$-complete if π is in $\mathcal{NP}$ and π is $\mathcal{NP}$-hard.

The following are examples of $\mathcal{NP}$-complete problems: SAT, CLIQUE, Node Cover Decision (or Vertex Cover) Problem, 3SAT, Subset Sum, Partition, Traveling Salesperson Problem, and Hamiltonian Cycle. We briefly summarized Cook's theorem that states that SAT is $\mathcal{NP}$-complete. Using this theorem we showed that the following problems are also $\mathcal{NP}$-complete: CLIQUE, Independent Set, Node Cover Decision Problem.