Problem Set 1—Due Tuesday , April. 19, 5PM

NOTE: please explain and justify your solutions clearly. Give a high-level overview before plunging into details (consider the lecture formats as a model)

(20) Problem 1.

Suppose we have two n by n matrices A,B and we want to compute $C = A \times B$, the normal matrix product. Note that the normal serial algorithm for this takes $O(n^3)$ time. Though there are fancier algorithms that are faster, you need not worry about matching their time (and thus work).

Describe an efficient and fast parallel CREW PRAM algorithm for this problem. You may refer to algorithms we have studied to help solve this. Analyze the work and Time to solve this and give the maximum number of processors your algorithm can use effectively.

(10) Problem 2.

Prove that no EREW algorithm can solve the prefix-sum problem on n values in better than O($logn$) time using $p = n$ processors (thus the algorithm we gave is as fast as possible for thi setting). You may use the fact that an EREW algorithm take takes an array of n bits as input and outputs the OR of those bits requires $\Omega(logn)$ time using $p = n$ processors.

(15) Problem 3. Exercise 2 in the notes, page 21.

(15) Problem 4. Exercise 4, problem (1), page 22.

(20) Problem 5. Modify the selection algorithm described in chapter 5 (the reduction one) so it works even if there are duplicate values in the list. You should maintain the same time and work bounds. Also, for a best solution you should return not just the value which is the kth in the sorted list, but the actual position in the original input that matches the kth largest (that is, if $k = 100$, and we have 5 identical values that are tied for rank $98, 99, 100, 101$ in positions $30, 37, 120, 171$ in the input array, you should return 120, the position that would mapped to position 100 of the sorted output if we use a stable sort. Note, you may assume that the parallel sorts being used are stable.