38. (2 points) We consider the problem of multiplying two n by n matrices. Assume that the sequential time algorithm that you wish to compare to has time complexity $S(n) = n^3$.

- Design a parallel algorithm that runs in time $O(n)$ on a CREW PRAM with n^2 processors.
  - What is the efficiency of this algorithm?
  - Using the folding principle, what upper bound would you get on the running time for this algorithm on $n^{1/4}$ processors?
- Design a parallel algorithm that runs in time $O(\log n)$ time on a CREW PRAM with n^3 processors.
  - What is the efficiency of this algorithm?
  - Using the folding principle, what upper bound would you get on the running time for this algorithm on $n^{1/4}$ processors?
- Design a parallel algorithm that runs in time $O(\log n)$ time on a CREW PRAM with $n^3/\log n$ processors.
  - What is the efficiency of this algorithm?
  - Using the folding principle, what upper bound would you get on the running time for this algorithm on $n^{1/4}$ processors?
- Design a parallel algorithm that runs in time $O(\log n)$ time on a EREW PRAM with $n^3/\log n$ processors. .
  - What is the efficiency of this algorithm?
  - Using the folding principle, what upper bound would you get on the running time for this algorithm on $n^{1/4}$ processors?

39. (4 points) Consider an directed acyclic graph $G = (V, E)$ where the vertices V are operations. A directed edge (v, w) means that operation v has to be completed before operation w can begin. You can assume without loss of generality that the edge relationship is transitive. You can assume that if there is not an edge between vertices v and w that these two operations can be executed in parallel. Assume that each operation takes unit time. Assume that our unit of time is sufficiently large so that context switching delays between operations are negligible. Let n be the number of vertices in G. Let L be the number of vertices in the longest path in G. Give an polynomial time algorithm to schedule the operations in V and prove that the time used by the schedule produced by your algorithm is $O(L + n/p)$ assuming p processors.

40. (6 points) We consider the problem of adding two n bit integers.

(a) Give an algorithm that runs in $O(\log n)$ time on a CREW PRAM with n processors.
- NOTE: If your algorithm is EREW, you might want to rethink since I don't know how to do this easily without CR.
- HINT: Use divide and conquer and generalize the induction hypothesis.

(b) Give an algorithm that runs in $O(\log^2 n)$ time on an EREW PRAM with n processors.

41. (8 points) Explain how to solve the longest common subsequence problem in time $O(\log^2 n)$ using at most a polynomial number of processors on a CREW PRAM.

HINT: One way to do this is to reduce the longest common subsequence problem to a shortest path problem.