1. Consider the problem of computing the AND of n bits.

   - Give an algorithm that runs in time $O(\log n)$ using n processors on an EREW PRAM. What is the efficiency of this algorithm?
   - Give an algorithm that runs in time $O(\log n)$ using $n/\log n$ processors on an EREW PRAM. What is the efficiency of this algorithm?
   - Give an algorithm that runs in time $O(1)$ using n processors on a CRCW PRAM.

   †

2. You know that lots of famous computer scientists have tried to find a fast efficient parallel algorithm for the following Boolean Formula Value Problem:

   INPUT: A Boolean formula F and a truth assignment A of the variables in F.

   OUTPUT: 1 if A makes F true, and 0 otherwise.

   Moreover, most computer scientists believe that there is no fast efficient parallel algorithm for the Boolean Value Problem. You want to find a fast efficient parallel algorithm for some new problem N. After much effort you can not find a fast efficient parallel algorithm for N, nor a proof that N does not have a fast efficient parallel algorithm. How could you give evidence that finding a fast efficient parallel algorithm for N is at least a hard of a problem as finding a fast efficient parallel algorithm for Boolean Formula Value problem? Be as specific as possible, and explain how convincing the evidence is.

   Note that "fast and efficient" means poly-log time with a polynomial number of processors. The term "poly-log" means bounded by $O(\log^k n)$ for some constant k.

   ††

3. Consider the problem of taking as input an integer n and an integer x, and creating an array A of n integers, where each entry of A is equal to x.

   - Give an algorithm runs in time $O(\log n)$ on a EREW PRAM using n processors. What is the efficiency of this algorithm?
   - Give an algorithm that runs in time $O(\log n)$ on a EREW PRAM using $n/\log n$ processors. What is the efficiency of this algorithm?
   - Give an algorithm that runs in time $O(1)$ on a CRCW PRAM using n processors. What is the efficiency of this algorithm?

   †

4. Design a parallel algorithm for the parallel prefix problem that runs in time $O(\log n)$ with $n/\log n$ processors on a EREW PRAM.

   ††

5. Give an algorithm that given an integer n computes $n!$, that is n factorial, in time $O(\log n)$ on an EREW PRAM with n processors. Make the unrealistic assumption that a word of memory can store arbitrarily large integers.

   †

6. We consider the problem of multiplying two n by n matrices.

   - Design a parallel algorithm that runs in time n on a CREW PRAM with n^2 processors. What is the efficiency of this algorithm?

- 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?
- 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?
- 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? HINT: Recall problem 3.

††

7. Design a parallel algorithm that given a polynomial $p(x)$ of degree n and an integer k computes the value of $p(k)$. You algorithm should run in time $O(\log n)$ on a EREW PRAM with $O(n/\log n)$ processors. Assume that the polynomial is represented by its coefficients.

   ††

8. We consider the problem of computing F_n, the nth Fibonacci number, given an integer n as input. Show how to solve this problem in time $O(\log n)$ on a EREW PRAM with n processors. Make the unrealistic assumption that F_n will fit within one word of memory for all n, that is, assume that all arithmetic operations take constant time. Recall that F_n is defined by the following recurrence: $F_0 = F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n > 1$.

   HINT: Note that for $j > 0$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_j \\ F_{j-1} \end{bmatrix} = \begin{bmatrix} F_{j+1} \\ F_j \end{bmatrix}$$

   ††

9. The input to this problem is a character string C of n letters. The problem is to find the largest k such that

$$C[1]C[2]\ldots C[k] = C[n-k+1]\ldots C[n-1]C[n]$$

   That is, k is the length of the longest prefix that is also a suffix. Give a EREW parallel algorithm that runs in poly-logarithmic time with a polynomial number of processors.

   ††

10. The input to this problem is a character string C of n letters. The problem is to find the largest k such that

$$C[1]C[2]\ldots C[k] = C[n-k+1]\ldots C[n-1]C[n]$$

    That is, k is the length of the longest prefix that is also a suffix. Give a CRCW parallel algorithm that runs in constant time with a polynomial number of processors.

    ††

11. Design a parallel algorithm for adding two n bit integers. You algorithm should run 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 with out CR.

    † † †

    HINT: Use divide and conquer and generalize the induction hypothesis.

12. Explain how to modify the all-pairs shortest path algorithm for a CREW PRAM that was given in class so that it runs in time $O(\log^2 n)$ on a EREW PRAM with n^3 processors.

    †

13. Explain how to modify the all-pairs shortest path algorithm for a CREW PRAM that was given in class so that it actually returns the shortest paths (not just their lengths) in time $O(\log^2 n)$ on a EREW PRAM with n^3 processors.

    ††

14. 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.

    † † †

15. Give an algorithm for the minimum edit distance problem that runs in poly-log time on a CREW PRAM with with a polynomial number of processors. Here poly-log means $O(\log^k n)$ where n is the input size, and k is some constant independent of the input size.

    Recall that the input to this problem is a pair of strings $A = a_1 \ldots a_m$ and $B = b_1 \ldots b_n$. The goal is to convert A into B as cheaply as possible. The rules are as follows. For a cost of 3 you can delete any letter. For a cost of 4 you can insert a letter in any position. For a cost of 5 you can replace any letter by any other letter.

    † † †

16. Design a parallel algorithms that merges two sorted arrays into one sorted array in time $O(1)$ using a polynomial number of processors on a CRCW PRAM.

    † † †

17. Design a parallel algorithm that finds the maximum number in a sequence $x_1, \ldots, x_n$ of (not necessarily distinct) integers. Your algorithm should run in time $O(\log \log n)$ on a CRCW PRAM with n processors.

    HINT: Recall that you can find the maximum of k numbers in $O(1)$ time with $\Omega(k^2)$ processors. Apply this fact repeatedly.

    † † †

18. Design a parallel algorithm that finds the maximum number in a sequence $x_1, \ldots, x_n$ of (not necessarily distinct) integers in the range 1 to n. Your algorithm should run in constant time on a CRCW PRAM with n processors. Note that it is important here that the x_i's have restricted range.

    † † †

19. Give a parallel algorithm for the following problem that runs in time $O(\log n)$ on an EREW PRAM. The input is a binary tree with n nodes. Assume that each processor has a pointer to a unique node in the tree. The problem is to number the leafs consecutively from left to right (that is in-order). Note that this algorithm is needed for the algorithm in the notes for computing arithmetic expressions.

    ††

20. Design a parallel algorithm that takes a binary expression tree, where the leaves are integers, and the internal nodes are the four standard arithmetic operators addition, subtraction, multiplication, and division, and computes the value of the expression. Your algorithm should run in $O(\log n)$ time on a CREW PRAM with n processors, where n is the number of nodes in the tree. You may assume that each processor initially has a pointer to a unique node in the tree.

    HINT: Following the technique used for subtraction in the class notes you need only find a collection of functions that contain the identity function and constant functions, and is closed under addition, subtraction, multiplication, division with constants, and composition.

    † † †

21. Give a parallel algorithms for the following problem that runs in time $O(\log n)$ on an EREW PRAM with n processors. The input is a fully paranthesized arithmetic expression, with n symbols, stored in an n element array in the standard in-order fasion. The output should be an expression tree with each processor having a pointer to a unique node in the tree.

    NO HINT: This is the one problem assigned this term that I don't know how to do. This was from an old text that we used to use. I seem to recall that the text said the problem was easy, but I worked on it for at least 15 minutes and couldn't solve it, so decided to assign it as homework several years ago. No one has solved it yet.

† † ††