Problem Set 1 for CS 170

Problem 0

The first question of each homework will be on opportunity for you to provide us some feedback. In fact, it will be a requirement. Although you may not have much to say, please say something. This problem will be graded. This week, we'd like to know what's the one thing you'd like to see explained better in lecture or discussion sections? (Sometimes we botch the description of some concept, leaving some people confused. Sometimes we omit things people would like to hear about. Sometimes the notes are very confusing on some point.)

Problem 1

On most computers, the operations of subtraction, testing the parity (odd or even) of a binary integer, and halving can be performed more quickly than computing remainders. This problem investigates the *binary gcd algorithm*, which avoids the remainder computations used in Euclid's algorithm.

(a) Prove that if a and b are both even, then $\gcd(a, b) = 2\gcd(a/2, b/2)$.

(b) Prove that if a is even and b is odd, then $\gcd(a, b) = \gcd(a/2, b)$.

(c) Prove that if a and b are both odd, then $\gcd(a, b) = \gcd(|a - b|/2, b)$.

(d) Design an efficient binary gcd algorithm for input integers a and b, where $a \geq b$, that runs in $O(\log^2 a)$ time. (assume that subtracting two n bit integers takes $O(n)$ steps, and testing parity and halving can be performed in unit time).

Problem 2

Let's compute the Fibonacci series in yet another way. Consider the following recursion:

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$$

with

$$\begin{pmatrix} F_1 \\ F_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

(a) Based on this recursive definition, write $\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix}$ in terms of $\begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$.

(b) How many matrix multiplications are required to calculate this expression?

(c) What is the bit complexity of the algorithm if binary multiplication is n^2?

(d) How fast do we have to make binary multiplication in order for this algorithm to be faster than the one given in class?

Problem 3

(a) Prove or disprove: If a has an inverse modulo b, then b has an inverse modulo a.

(b) Prove or disprove: If $ax \equiv bx \bmod c$, then $a \equiv b \bmod c$.

(c) If p is prime, how many elements of $\{0, 1, ..., p^n - 1\}$ have an inverse modulo p^n?

Problem 4

The algorithm for computing $a^x \pmod N$ by repeated squaring does not necessarily lead to the minimal number of multiplications. Give an example x ($x > 10$) where the exponentiation can be performed using fewer multiplications using some other method.

BONUS!

In class we saw that in order to implement a shallow (depth $\log n$) circuit for adding two numbers, it is crucial to be able to quickly determine the carry bit in each position. Say that the two n-bit numbers being added are $a_{n-1}...a_0$ and $b_{n-1}...b_0$. Define the carry propogate bit $p_i = a_i + b_i$, and the carry generate bit $g_i = a_i \cdot b_i$. Let d_i denote the actual carry into the ith position. Then $d_i = d_{i-1} \cdot p_{i-1} + c_{i-1}$. We wish to compute all of the d_i's efficiently in parallel. Let $A_i = \begin{pmatrix} p_i & c_i \\ 0 & 1 \end{pmatrix}$ and $M_i = A_{i-1}A_{i-2}...A_1A_0$, where we use boolean matrix multiplication (i.e. $+$ is replaced by OR, and $\cdot$ is replaced by AND).

(a) Show how to find d_i from M_i.

(b) Now we would like to find all of the M_i's. If we computed them sequentially (i.e. $M_i = A_{i-1}M_{i-1}$), then we would require $O(n)$ steps. Describe how to do the computation in parallel using only $O(\log n)$ steps.

HINT: Group the matrices in pairs (e.g. $B_1 = A_2A_1$, $B_2 = A_3A_4$, ...) and solve this subproblem of size $n/2$ whose output is $n/2$ matrices. Now give a procedure for quickly computing the n matrices, M_i, in parallel from these $n/2$ matrices. Where do you use the fact that boolean matrix multiplication is an associative operation?