CIS 667 and CIS 467H: Homework 5(Due: Apr 21, 2005)

Problem 1. (20 points)

You are given an a-bit positive integer A and an b-bit positive integer B. You may assume $A \geq B$. How fast can you compute $A+B$, $A-B$, $A \cdot B$, and find Q, R such that $A = BQ + R$, $0 \leq R < B$? Express your answer in terms of a and b and justify it by pointing to the appropriate reference or give the algorithm. You can cite results of the notes/textbook covered in class.

Problem 2. (20 points)

(a) Fibonacci Revisited. Show that F_n can be computed in $O(n^2)$ bit operations even if n-bit multiplication can only be done in $\Theta(n^2)$ bit operations.

(b) Suppose you want to raise intefer x to the y power i.e. computer $n = x^y$. Show that this requires $O(\lg^2 n)$ bit operations.

Problem 3. (20 points)

The prime number theorem states that the number of primes less than x is about $x/\ln x$. Therefore the density of primes is $1/\ln x$. In other words if we pick uniformly at random an integer z between 1 and N with probability $1/\ln N$ we choose a prime number and with probability $1 - 1/\ln N$ we choose a composite number. Let $A = A_m \ldots A_1$ be an m-bit integer. Let $B = B_m \ldots B_1$ be an m-bit integer. Then $A = A_m 2^{m-1} + \ldots + A_1$ and $B = B_m 2^{m-1} + \ldots + B_1$.

(a) Is $A \leq 2^m$? Explain. Is $B \leq 2^m$? Explain (3 points).

(b) If $p/(A-B)$ then show that $A \bmod p \equiv B \bmod p$. Then show that if $A \bmod p \equiv B \bmod p$, the $p/(A-B)$.

(c) Show A or B can have at most m distinct prime divisors.

(d) Let us pick a random number prime number p between 1 and $4n^2 m \ln n$, where $n > m$. Show that such a p divides $A - B$ with probability at most $O(1/n^2)$.

Problem 4. (10 points)

In Rabin-Karp let A be the pattern P of length m and T be the text of length n.

This is a continuation of Problem 2. Choose R so that $R = 4n^2 m \ln(4n^2 m)$; R is the prime number used to perform fingerprint computations modulo R.

(a) Show that the probability a spurious hit occurs in position i of the text is $O(1/n^2)$.

(b) Show that the probability a spurious hit occurs anywhere in the text is $O(1/n)$.

(c) Show that the expected running time of the algorithm is $O(n)$.

Problem 5. (20 points)

Show that the GCD computation requires $O(\lg a \lg b)$ bit operations thus improving the rough $O(\lg^3 a)$ bound given in class, i.e. complete the proof of Corollary 5 of the notes.

Hint. Use Corollary 4. Be very careful in counting operations. Go through all the division steps and count the bit operations individually and be careful not to overestimate them. Treat the first division step separately.

Problem 6. (10 points)

In Homework 4, perfect power was analyzed in the word model (one multiplication cost one operation). Analyze the algorithm in the bit model and show that you can determine whether A is a perfect power or not in $O(\lg^k A)$ bit operations. Keep k strictly) smaller than 4. Note that you count bit operations now and the only input you are given is A in binary.

CIS 667 and CIS 467H: Homework 5(Due: Apr 21, 2005)

Option 1. (100 points)

Part A (80 points)

Implement the algorithm for the perfect power problem (see Solutions of Homework 4 also) in C or C++.However A can be arbitrarily long (eg. 1024 or 8192 bits). You are allowed to use libraries for arbitrary precision arithmetic as long as (a) they are for free (i.e. I can use them and install them on a linux machine to test your code) and (b) they are easily installable (i.e. I can install them easily). Alternatively, you can implement your own functions for auxiliary operations (eg. arbitrarily long multiplication, exponentiation, etc).

I expect as an answer a .tar file sent by email. The .tar will be untarred on a Red Hat linux workstation and compiled through gcc. I expect a make install command to compile everything and create an executable file named powertest. powertest will accept as input a file containing n in decimal notation.

Thus if file myfile contains a base-10 integer such as

```
17487686712733928413644063750880864826316326531082890839798047131393
06920507121153268532108269241880671097659463948848837902966613039193
61801624726151915125942668649508993665419986623407409256320591797254
65901781768127877188903984695217669171609482765052925389918678373962
03339268758977282743385306120289869213112851446870454351518286400633
04862801177174173384780775389699560332291136389670468588940721006781
36076427022136733286307364152390825026213339153406940132529505642288
77265570344122667987101745048846965145 6
```

then the following command should return

```
% powertest myfile
  x= 123456 y=101
  It's a power!
```

within a reasonable amount of time (eg. under 60 seconds for integers as long as 8192 bits, i.e with up to 2000-3000 digits).

Part B (20 points)

For arbitrarily long numbers (refer to Part A for limitations), implement the GCD, the extended GCD (both non-recursive), and the modular equation solving algorithm.

Part C (20 points)

For arbitrarily long integers implement the Chinese Remainder Theorem. Based on the discussion in class, n can be arbitrarily long say up to 8192 bits, and maintain n_i relatively prime and up to 32 bits each (i.e. they can be represented by a C/C++/Java `int` data type). Provide operations add, subtract, multiply based on the Chinese Remainder Theorem, as explained in class.

You can choose to do Part B or Part C or do both to collect 20 extra points.