

CS 70 FALL 2007 — DISCUSSION #4

ASSANE GUEYE, LUQMAN HODGKINSON, AND VAHAB POURNAGHSHBAND

1. ADMINISTRIVIA

- (1) Course Information
 - **Reminder:** The first midterm will be held on **Wednesday October 3rd** 6pm-8pm in 10 Evans
- (2) Discussion Information
 - The third homework is graded and will be distributed in section.

2. MODULAR ARITHMETIC

In modular arithmetic, we concern ourselves with the notion of *congruences*. Much like equalities in normal arithmetic, congruences form a so-called *equivalence relation*. That is, they satisfy the following three properties:

- (1) Reflexive: $a \equiv a \pmod n$
- (2) Symmetric: $a \equiv b \pmod n \implies b \equiv a \pmod n$
- (3) Transitive: $a \equiv b \pmod n, b \equiv c \pmod n \implies a \equiv c \pmod n$

Recall that when we're looking at numbers *modulo* n , (sometimes denoted as $\mathbb{Z}/n\mathbb{Z}$ and more succinctly as \mathbb{Z}_n),

$$a \equiv b \pmod n \iff n|(a - b)$$

Why is this true? Recall that if $a \equiv b \pmod n$, then this means that $a = b + nk$ for some $k \in \mathbb{Z}$. Then clearly we have $nk = a - b$, from which it follows that n must divide $a - b$.

Addition and multiplication in \mathbb{Z}_n work the same as they do in \mathbb{Z} . The following rules hold:

- (1) $a \equiv b \pmod n \implies a + c \equiv b + c \pmod n$
- (2) $a \equiv b \pmod n \implies a \cdot c \equiv b \cdot c \pmod n$

But what about subtraction and division? Well, subtraction is easy, since additive inverses always exist in \mathbb{Z}_n . For example, if you consider $a - b$ in $\mathbb{Z}/n\mathbb{Z}$, the quantity $-b$ is congruent to $n - b, 2n - b, 3n - b, \dots$. So you can always turn subtraction into addition¹. For instance, $-4 \equiv 3 \pmod 7$. Division, though, is not so easy. It turns out that you can't always divide by a number in modular arithmetic. Let's consider what it means when you want to solve for x in the equation

$$2x \equiv 6 \pmod 8.$$

Date: September 26, 2007.

¹This is somewhat imprecise language, but if you think of it in the following way it might be helpful: When you subtract, you're really just adding the additive inverse. So $a - b$ becomes $a + (-b)$.

Well, clearly $x \equiv 3 \pmod{8}$ is a solution. But $x \equiv 7 \pmod{8}$ is also a solution! So, in a sense, division by 2 in $\mathbb{Z}/8\mathbb{Z}$ is not well-defined (how can x be congruent to both 3 and 7?)

However, if we try and solve for x in the problem

$$3x \equiv 6 \pmod{8},$$

the only solution is $x \equiv 2 \pmod{8}$ (try it out!) Can you guess when you're allowed to divide and when you aren't?

Now let's move on to exercises:

Exercise 1. Show that if a is an odd integer, then $a^2 \equiv 1 \pmod{8}$.

Exercise 2. Show that if $n \equiv 3 \pmod{4}$, then n cannot be the sum of the squares of two integers.

Exercise 3. What is $2^{2^{2006}} \pmod{3}$?

Exercise 4. Prove: $\forall n \in \mathbb{N}, \text{ if } a \equiv b \pmod{m}, \text{ then } a^n \equiv b^n \pmod{m}$

3. ALGORITHM COMPLEXITY AND BIG-O NOTATION

Exercise 5. Rank the following functions according to asymptotic growth:

- (1) n^n
- (2) $2^n \log n$
- (3) $n \log n^2$
- (4) $\log 2^n$
- (5) \sqrt{n}
- (6) $\log n^2$.

Exercise 6. Find the tightest bound for the following algorithms using the big-O notation.

- (1) `find_max(list)`

```

temp = list[1];
for i=2 to length(list) do
    if list(i)>temp
        temp = list(i);
    end % if
end % for
return temp

```
- (2) `Mat_mult(A,b)`

```

c=zeros(m,1); % Initialize the output
for i=1 to m do
    temp=0;
    for k=1 to n do
        temp = temp+A(i,k)*b(k);
    end %for k
    c(i)=temp;
end % for i
return c

```
- (3) `Mat_mult(A,B)`

```

C=zeros(n,n); % Initialize the output
for i=1 to n do
    for j=1 to n do
        temp=0;
        for k=1 to n do
            temp = temp+A(i,k)*B(k,j);
        end %for k
        C(i,j)=temp;
    end % for j
end % for i
return C

```