

## Homework Assignment Two due Monday, February 11 in class.

Total 50 points. Every homework assignment will be 50 points total. After dropping the worst score, there will be three scores left with a maximum total of 150 points for 30% of your course grade. Remember that homework must be on flat (not spiral) 8 1/2 by 11 inch paper and must be neat. Staple together multiple pages. Homework must be turned in in the first fifteen minutes of class on Monday, February 11, 2008.

1. (10 points) In the last problem set, you showed that if both  $x$  and  $y$  are odd integers (not divisible by 2),

$$\gcd(x, y) = \gcd((x - y)/2, y)$$

Now show the following two statements are also correct:

- (a) If  $x$  and  $y$  are both even,

$$\gcd(x, y) = 2 \gcd(x/2, y/2)$$

- (b) If  $x$  is even and  $y$  is odd,

$$\gcd(x, y) = \gcd(x/2, y)$$

(A symmetric argument can be given if  $y$  is even and  $x$  is odd. You need not do this.)

Then (1) write pseudocode to use these formulas to give an alternative to the euclidean algorithm for finding the greatest common divisor of any two numbers. (Hint: the most interesting part is how you stop the recursion.) Last (2), give a reason why this new algorithm is also  $O(\log(\max(x, y)))$ . (Unlike the textbook, we consider the division of two numbers to be an  $O(1)$  operation. Thus we are only concerned with counting the number of recursive calls made in your algorithm.)

2. (10 points) Consider an RSA set as in Figure 1.9 on page 34. Let  $p = 17$  and let  $q = 19$ . Then  $N = (17)(19) = 323$ . Also,  $(p - 1)(q - 1) = (16)(18) = 288$ . Let  $e = 5$ . This choice is valid since 5 and 288 are relatively prime.

- (a) Find the secret key  $d$ , the inverse of  $e$  modulo  $(p - 1)(q - 1)$ .  
(b) Suppose the message is  $x = 999$ . Use Modexp (Figure 1.4, page 19) to find the encryption of the message. That is, find  $x^e \bmod N$ .

3. (10 points)

For this problem, you need to know the sum of a geometric series

$$\sum_{i=0}^k r^i = (r^{k+1} - 1)/(r - 1)$$

For each of the following pseudocode programs:

- (a) Draw the three top levels of a recursion tree (as on page 48 of the text). Show how many recursive calls are made at each level by how many children each node has. Thus if the program has three recursive calls, each node in the tree (except at the leaves) has three children.
- (b) Show the cost of each level of the tree you drew. Assume the cost of the operation "print" is  $c$ , and do not count the cost of checking if  $n$  is positive or the cost of making a recursive call. That is, only count the cost of the printing.
- (c) Tell how many levels there are in the whole tree and explain why. Do not worry about being off by one. For programs 2 and 4, you may assume  $n$  is a power of 2. For program 6, you may assume that  $n$  is a power of 3.
- (d) Give a function  $f(n)$  which expresses the total cost of the algorithm.
- (e) Tell if  $f(n)$  is in  $\Theta(\log(n))$ ,  $\Theta(n)$ ,  $\Theta(n \log n)$ ,  $\Theta(n^2)$ ,  $\Theta(2^n)$  or  $\Theta(3^n)$ . (For example,  $f(n) = 5n^2 + 47n + 32$  is in  $\Theta(n^2)$ .) You need not prove your assertion.

Prog1(n)

```
    if n > 0
        print "hello"
```

```

                Prog1(n - 1)
=====

Prog2(n)
    if n >0
        print "hello"
        Prog2(floor(n/2))
=====

Prog3(n)
    if n>0
        print "hello"
        Prog3(n - 1)
        Prog3(n - 1)
=====

Prog4(n)
    if n >0
        print "hello"
        Prog4(floor(n/2))
        Prog4(floor(n/2))
=====

Prog5(n)
    if n>0
        print "hello"
        Prog5(n - 1)
        Prog5(n - 1)
        Prog5(n - 1)
=====

Prog6(n)
    if n >0
        print "hello"
        Prog6(floor(n/3))
        Prog6(floor(n/3))
        Prog6(floor(n/3))

```

4. (10 points) In the famous Tower of Hanoi problem, there are three wooden pegs. A tower of disks with holes in the center is placed on

one peg. The disks are arranged so that if a disk  $A$  is above any disk  $B$ , then the diameter of  $A$  is smaller than the diameter of  $B$ . The problem is to move the entire tower, one disk at a time, to one of the other pegs, without ever having a larger diameter disk above a smaller diameter disk on the same peg. Give an analysis of this problem, showing how many moves must be made if you start with  $n$  disks on one peg. (No credit for answer alone). Hint: If  $n = 2$ , here is how you do it. Suppose the pegs are named P1, P2 and P3. The small disk is named S and the larger disk is named L. Suppose S and L are on P1 and you want to move the tower to P2. You must first move S to P3, then move L to P2, then move S to P2. Now work this out for  $n = 3$ . This should give you an idea for a recursive program. The problem is to calculate the cost of the recursive program. You should draw a recursion tree as in the previous problem.

5. (10 points) Do problem 2.23 on page 75 of the text. We should discuss this in class. This is the problem of finding out whether an array has a *majority element*, that is, a value which occurs in more than half of the array entries.