Math 4330, Homework 6, 2/24/2014 : Due 3/10/2014 [1]

Read 4.5.4 up to and including page 386.

1. Suppose N is an odd composite number and $S = \{(x, y) \in \mathbb{Z}^2 \ : \ x^2 \equiv y^2 \pmod{\mathrm{N}}\}$.
(i) Prove that if (u, v) is chosen randomly from S, there is at least a 50% chance that $u \not\equiv \pm v \pmod{\mathrm{N}}$.
(ii) If $(u, v) \in S$ and $u \not\equiv \pm v \pmod{\mathrm{N}}$, explain how these two integers can be used to find a proper divisor of N.

Solution: For (i), there's a small typo here, so I'll give everyone more time to work on it. It should read: "Suppose N is an odd composite number divisible by at least two distinct primes and ...".

For (ii), if $(u, v) \in S$ then $u^2 \equiv v^2 \pmod{\mathrm{N}}$, so that $(u - v)(u + v) \equiv 0 \pmod{\mathrm{N}}$. Therefore, $\gcd(u - v, N)$ and $\gcd(u + v, N)$ cannot both be 1. On the other hand, since $u \not\equiv \pm 1 \pmod{\mathrm{N}}$, it follows that $u + v \not\equiv 0 \pmod{\mathrm{N}}$ and $u - v \not\equiv 0 \pmod{\mathrm{N}}$, so that these gcd's cannot equal N either. Therefore, at least one of $\gcd(u - v, N)$, $\gcd(u + v, N)$ is a proper divisor of N (and in fact, they both are).

2. Write a function `trialDivision(n)` to find a proper divisor of each of the following numbers: $3^{15} + 40$, $3^{19} + 22$, $3^{25} + 16$, $3^{29} + 8$, $3^{31} + 26$, $5^{25} + 6$. Your program should also report the time required to factor each number, using the following code snippet:

```
Removed - it's in the solution below.
```

3. Use Algorithm B (omit Step B2 and include the suggestion on p. 386 to handle failures) to find divisors of the same numbers from the previous exercise (it need not completely factor the numbers - just find a single proper divisor of each). Your program should also report the time required to factor each number. Compare the timing results from this and the previous exercise side-by-side in a table.

Solution for Problems 2 and 3: The following function does the trick:

```
# HW6, Chris Monico, 4/1/2014
import datetime
##################
def gcd(u,v):
    if u==0:
        return v
    if u<0:
```

```
        u *= -1
    if v<0:
        v *= -1
    return gcd(v%u, u)
##################
def trialDivision(n):
    start = datetime.datetime.now()
    d=n
    i = 2
    if n <= 1: return n
    while i <= n:
        if n%i == 0:
            d=i
            break
        i = i + 1
    stop = datetime.datetime.now()
    elapsed = stop - start
    elapsedSeconds = elapsed.seconds + (elapsed.microseconds/1000000.0)
    print "trialDivision(): elapsed time %f seconds. " % (elapsedSeconds)
    return d
###################
def pollardRho(N):
    start = datetime.datetime.now()
    x=5
    xprime=2
    k=1
    ell=1
    n=N
    # We will allow different choices of c to accommodate the
    # (rare) possibility of failure.
    c=1
    divisorFound = False
    while not divisorFound:
        g = gcd(xprime-x, n)
        while g==1:
            k -= 1
            if k==0:
                xprime = x
                ell = 2*ell
                k = ell
            x = (x*x+c)%n
            g = gcd(xprime-x, n)
        if g<n:
            divisorFound = True
```

```
        else:
            c += 1 # Try again with a different value of c.
    stop = datetime.datetime.now()
    elapsed = stop - start
    elapsedSeconds = elapsed.seconds + (elapsed.microseconds/1000000.0)
    print "pollardRho(): elapsed time %f seconds. " % (elapsedSeconds)
    return g
################
numbers=[3**15+40, 3**19+22, 3**25+16, 3**29+8, 3**31+26, 5**25+6]
for N in numbers:
    print trialDivision(N)
    print pollardRho(N)
```

Running this code on the given numbers, we obtain the following timings (on my machine - it will certainly be different on your machine):

N	divisor	$t_{\text{trialDivision}}$	$t_{\text{pollardRho}}$
$3^{15}+40$	409	0.000129	0.000654
$3^{19}+22$	20161	0.001392	0.002258
$3^{25}+16$	197767	0.058555	0.006828
$3^{29}+8$	1373233	0.363805	0.036486
$3^{31}+26$	5490391	1.422398	0.026401
$5^{25}+6$	218161103	59.525772	0.334806

4. Among all composite numbers of the form $N = pq$ with p and q prime and $\sqrt{N}/2 \leq p \leq 2\sqrt{N}$, roughly what is the largest such number that each of your programs could compute if they ran for one year nonstop? You should turn in a written solution with experimental results and a carefully reasoned argument supporting your conclusion.

Solution: In principle, we could use the known order of magnitude of the runtimes and then least-squares-fit the parameters. But since we're looking only for a rough estimate, we'll use the timings for the largest number factored and assume they're the most accurate.

Trial division required about 60 seconds to find the divisor 218161103. On the other hand, we know that it needs about p operations to find the prime divisor p, so we did about $(2.2 \times 10^8)/60 \approx 3.7 \times 10^6$ operations per second. There are about 3.2×10^7 seconds in a year, so the code could do about $(3.2 \times 3.7)10^{6+7} \approx 1.2 \times 10^{14}$ operations in one year. This means that in one year, the largest prime divisor p it could find would be about 1.2×10^{14}, and so the largest number of this form it could factor would have about $\boxed{28 \text{ digits.}}$

Pollard's rho method needs to do about $\sqrt{p}$ iterations to find the prime divisor p. It found the divisor 218161103 in about 0.334806 seconds, so the code is doing about

$\sqrt{2.2 \times 10^8}/0.335 \approx 4.5 \times 10^4$ iterations per second. So in one year, it could do about $(3.2 \times 10^7)(4.5 \times 10^4) \approx 1.4 \times 10^{12}$ iterations, which would allow it to find a prime divisor of about 24 digits. So the largest number of the given form that it could factor in a year would probably have about $\boxed{48 \text{ digits.}}$