Math 640:348 Prof. Kontorovich Spring 2015, 5/6 review session 2

We review factoring by the Pollard rho method

Take a composite integer, say

```
In[1]:= p = Prime[13];
        q = Prime[20];
        n = p q
Out[3]= 2911
```

We will iterate a "random" function, like

```
In[4]:= f[x_] := Mod[x^2 + 5, n];
```

and test when the tortoise has caught the hare by checking if gcd(y_i-x_i , n) > 1. We begin with

```
In[5]:= i = 0; x = 1; y = 1;
```

and apply f

```
In[6]:= i = 1;
        {x, y} = {f[x], f[f[y]]}
Out[7]= {6, 41}
```

What is a "collision" now? It's when x=y(mod p) (or mod q). But we don't know what p and q are! So we test whether

```
In[8]:= GCD[x - y, n]
Out[8]= 1
```

is non-trivial.

Now iterate

In[9]:=
```
i++;
{x, y} = {f[x], f[f[y]]}
GCD[x - y, n] ≠ 1
```

Out[10]= {41, 1465}

Out[11]= False

In[12]:=
```
i++;
{x, y} = {f[x], f[f[y]]}
GCD[x - y, n] ≠ 1
```

Out[13]= {1686, 1982}

Out[14]= False

In[15]:=
```
i++;
{x, y} = {f[x], f[f[y]]}
GCD[x - y, n] ≠ 1
```

Out[16]= {1465, 2112}

Out[17]= False

In[18]:=
```
i++;
{x, y} = {f[x], f[f[y]]}
GCD[x - y, n] ≠ 1
```

Out[19]= {823, 1178}

Out[20]= True

Aha! We found a collision. It's not a collision mod n, since x≠y, but it *is* a collision mod either p or q. Indeed,

In[21]:=
```
GCD[x - y, n]
```

Out[21]= 71

So 71 is a factor of n, the other one being

In[22]:=
```
n / 71
```

Out[22]= 41

Note that, had we known p, q from the beginning, this really would be a genuine collision:

In[23]:=
```
Mod[{x, y}, p]
```

Out[23]= {3, 30}

Not mod p

In[24]:=
```
Mod[{x, y}, q]
```
Out[24]=
```
{42, 42}
```

But yes mod q; that's the real collision. But since we shouldn't know p or q to start, the way to test for this collision is as above, namely, by checking whether gcd[x-y,n]>1.

How long did we expect to run the algorithm? Roughly n^1/4 =

In[25]:=
```
n^(1/4) // N
```
Out[25]=
```
7.34532
```

steps, and we halted in

In[26]:=
```
i
```
Out[26]=
```
5
```

steps. Pollard rho wins again.