Pollard-Rho Factoring Method

n = pq

2, 5, 26, … mod p

2, 5, 26… mod q

2, 5, 26 … mod n

Eventually, the sequence must repeat…

If sequence repeats for mod p, when we look at the sequence mod n, it will not have repeated at that point, but the spaced out values will share a common factor of p, which we can discern via Euclid's Algorithm.

2, 8, 13, 19, 16, 1, 2, 8, 13, 19, 16, 1, 2, 8, 13

Compare 1^{st} 2^{nd}

Compare 2^{nd} and 4^{th}

Compare 3^{rd} and 6^{th} term

Pollard Rho says

Do 1^{st} and 2^{nd}

Do 2^{nd} and 4^{th}

Do 4^{th} and 8^{th}

Do 8^{th} and 16^{th}etc.

A = 2, 4, 16, 256,…, 2^1, 2^2, 2^4, 2^8, 2^{16}

B = 2, 16, 2^{16}, 2^{64}

A = 2

B = 2,

```
While (true) {
    A = ((a*a) +1) mod n
    B = ((b*b) + 1) mod n
    B = ((b*b) + 1) mod n
}
```

Fast Modular Exponentiation

For large integers a, b and c, calculate a^b mod c

Normal method:

```
int ans = 1;
for (int i=0; i<b; i++)
     ans = (ans*a)%c
```

Number of Steps = roughly b multiplications and mods, O(c), or order of the exponent.

How do we speed this up???

Think about what Pollard Rho does, if I know a^2 mod c, then in one step I can get to a^4 mod c by multiplying a^2 by itself…we can build up to a pretty high exponent fast, by squaring our previous answer. (Bottom Up View)

Alternate Top Dop Down – Recursive View

If my exponent b, is even, why not calculate $a^{b/2}$ mod c first, and then just square it, instead of multiplying a over and over again.

$17^{1000000}$ mod 123457, why not calculate 17^{500000} mod 123457, and then just square that result because

$17^{1000000} = (17^{500000})^2$ (even case)

$17^{999999} = 17^{999998}$ x 17 (odd case)

```
Int fastmodexpo(int base, int exp, int mod) {
    If (exp == 0) return 1%mod;
    If (exp%2 == 0) {
        Int temp = fastmodexpo(base, exp/2, mod);
        Return (temp*temp)%mod;
    }
    Return (fastmodexpo(base,exp-1,mod)*base)%mod;
}
```

Nice Result I will Prove for you

A prime number p has exactly $\phi(p-1)$ primitive roots.

Quiz 4

Same Procedure as before
Part A: Euler Phi Function, Fermat's Theorem, Euler's Theorem, Discrete Log Problem
Part B: Miller-Rabin, Fermat Factoring, Pollard-Rho and Fast Modular Exponentiation, I will have one coding coding question somewhere.