Recall: Euclidean algorithm:

$a = q_1 b + r_1$

$b = q_2 r_1 + r_2$

$\dots$

$r_{n-3} = q_{n-1} r_{n-2} + r_{n-1} = \gcd(a,b)$

$r_{n-2} = q_n r_{n-1} + 0$

Proposition: $r_{i+2} \le \frac{1}{2} r_i$.

Corollary: $n \le 2k$.

Proof: $1 \le r_{n-1} \le \frac{1}{2} r_{n-3} \le \frac{1}{2^2} r_{n-5} \le \dots \le \frac{1}{2^{\lfloor \frac{n}{2} \rfloor}} r_{\underbrace{n-1-2\lfloor \frac{n}{2} \rfloor}_{-1 \text{ or } 0}}$

$\le \frac{1}{2^{\lfloor \frac{n}{2} \rfloor}} \cdot 2^k$

$\Rightarrow 2^{\lfloor \frac{n}{2} \rfloor} \le 2^k \Rightarrow \lfloor \frac{n}{2} \rfloor \le k \Rightarrow n \le 2k.$ $\boxtimes$

Recall: Complexity of EA is $O(nk^2)$

Therefore it is $O(k^3)$ whis is polynomial time.

§13.3. Taking powers modulo m.

We are given $a, b, m \in \mathbb{Z}^+$ of at most k bits ($\le 2^k$). Want to compute $a^b \pmod m$.

Naive method: start with a, then multiply the result by a $b-1$ times reducing modulo m each time.

It requires $b-1$ multiplications $\sim 2^k$ – not polynomial time.

Successive squaring:

Let $b = (b_{k-1} b_{k-2} \dots b_1 b_0)_2$

(a) Start with a. Take successive squares, each time reducing modulo m to compute the sequence

$$a^{2^0}, a^{2^1}, a^{2^2}, \dots, a^{2^{k-1}} \pmod m.$$

(b) Multiply those elements of this sequence which correspond to $b_i = 1$, each time reducing modulo m.

Step (a) requires $\leq k-1$ multiplications and reductions mod m, each of them takes $O(k^2)$ bit operations. In total it takes $O(k^3)$ bit operations.

The same estimate is true for step (b)

$\Rightarrow$ the complexity of successive squaring is $O(k^3)$ bit operations – polynomial time.

§13.4. Checking primality.

We are given n ($\leq k$ bits) and want to find out whether it is prime or not (do not need the factorization of n),

Theorem: (Agrawal - Kayal - Saxena, 2002)
There is a polynomial time algorithm which determines whether given n is prime.

Naive approach (trial division): Try small numbers $2 \le d \le \sqrt{n}$ as possible divisors of n. If for some d, $d | n$ then n is composite. Otherwise it is prime.

It requires up to $\sqrt{n} \sim 2^{k/2} = (\sqrt{2})^k$ – not polynomial time.

Better (faster) approach:

(a) Pick random a from $\{1, 2, \dots, n-1\}$.

(b) Compute $\gcd(a, n)$

(c) If the result is not 1 then n is composite

(d) If the result is 1, compute $a^{n-1} \pmod{n}$

(e) If the result is not 1, n is composite.

(f) If the result is 1 – ? (Choose different a and go back to (b)).

Definition. Let $1 < a < n$. Number n is called <u>pseudoprime for the base a</u> if
$$a^{n-1} \equiv 1 \pmod{n}$$
and n is composite.

Example: $n = 341 = 11 \cdot 31$ is pseudoprime for the base 2.

$$\left.\begin{array}{l} 2^5 = 32 \equiv 1 \pmod{31} \Rightarrow 2^{10} \equiv 1 \pmod{31} \\ 2^{10} \equiv 1 \pmod{11} \text{ by FLT} \end{array}\right\}$$

$$\Rightarrow 2^{10} \equiv 1 \pmod{341} \Rightarrow 2^{340} \equiv 1 \pmod{341}$$

(341 is not pseudoprime for the base 3)

Definition: Composite n is called a <u>Carmichal number</u> if $a^{n-1} \equiv 1 \pmod{n}$ for any $a \in \mathbb{Z}$ which is coprime with n.

Example: $n = 561 = 3 \cdot 11 \cdot 17$ is Carmichael number. – Ex.

In the worst case our faster approach is not better than trial division.