Outline

An efficient algorithm for Euclidean gcd; an efficient algorithm for modular exponentiation. An efficient algorithm for finding inverses in Z/N*.

- What does it mean for these algorithms to be efficient? Poly(log(N)); the number of bits required to express the number.

Analyze Euclidean Algorithm

Find $\gcd(a,b)$ when $a \geq b$

$a = bq + r_1 \qquad 0 \leq r_1 < b$

$b = r_1 q_1 + r_2 \qquad 0 \leq r_2 < r_1$

$r_1 = r_2 q_2 + r_3 \qquad 0 \leq r_3 < r_2$

$r_2 = r_3 q_3 + r_4 \qquad 0 \leq r_4 < r_3$

$\vdots$

$r_k = 0$

How many steps to reach $r_k = 0$

Observation: $\leq b$ steps of divisions

Running Time is $O(b) = O(2^{\log b})$ exponential w.r.t number of bits of b.

If $r_{i+1} \leq \frac{r_i}{2}$ for all i then running time is $O(\log b)$ linear w.r.t number of bits of b.

Is $r_{i+1} \leq r_i/2$? Not really.

If $r_{i+1} > \frac{r_i}{2}$,

$$r_i = q_{i+1} r_{i+1} + r_{i+2}$$

What could be the value of q_{i+1}?

Can q_{i+1} be 2?

NO. If $q_{i+1} = 2$ then $r_i = 2\underline{r_{i+1}} + r_{i+2}$

$> r_i + r_{i+2}$

not possible.

So $q_{i+1} = 1$

$r_i = r_{i+1} + r_{i+2}$

$\underline{r_{i+2}} = r_i - r_{i+1} < \underline{\frac{r_i}{2}}$.

At most two steps is required to reduce the value r_i to be by half.

In general, we can prove that

$$r_{i+2} < \frac{r_i}{2} \quad \text{for all values of } i$$

$$r_3 < \frac{r_1}{2} < \frac{b}{2}$$

$$r_5 < \frac{r_3}{2} < \frac{b}{4}$$

$$r_7 < \frac{r_5}{2} < \frac{b}{8}$$

$$r_{2k+1} < \frac{b}{2^k}$$

let $k = \lceil \log_2 b \rceil$

Then $\frac{b}{2^k} < 1$

$$r_{2k+1} < \frac{b}{2^k} < 1$$

So, $r_{2k+1} = 0$

After $2k+1 = 2\lceil \log_2 b \rceil + 1$ steps,

the algorithm terminates.

# of bits required to store an input $N = \log_2 N$

Analyze an algorithm : running time in terms of # of bits of N

$N = 2^k$, number of bits is k

$O(k^c)$ = polynomial

$O(k)$ = linear

$O(k^2)$ = quadratic

$O(e^k)$ = exponential

Efficient : polynomial

: