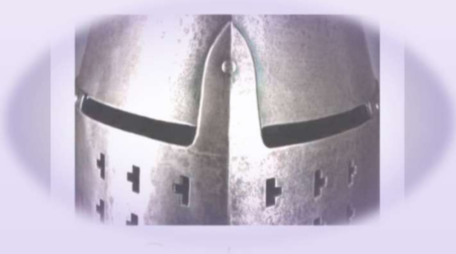


Chapter 2

Introduction to Number Theory

Divisibility

- $b|a$: b divides a, where a and b are integers
  - if $a = mb$ for some integer m
  - b is a divisor of a
- If $a|b$ and $b|c$, then $a|c$
- If $a|1$, then $a = \pm 1$
- $b|0$ for any $b \neq 0$
- If $a|b$ and $b|a$, $then\ a = \pm b$
- If $b|g$ and $b|h$, then $b\ |(mg + nh)$ for any integers m and n

Division algorithm

- divide a by $n, n > 0$
  - $\boldsymbol{a = qn + r}$
  - quotient: $q = \lfloor a/n \rfloor$
  - Remainder (residue) : $r = a - qn, 0 \leq r < n$
- $\lfloor x \rfloor$: the largest integer less than or equal to x

Modular arithmetic

- $a \bmod n = r, \;\; \mathrm{n} > 0$
  - n: the modulus
  - $r = a - qn$: the modular residue, where $q = \lfloor a/n \rfloor$
  - $0 \leq r < n$
  - $n|(a - r)$
- Examples
  - $25 \;\bmod 7 = 4, \;$ where $q = 3$
  - $-11 \bmod 7 = 3, \;$ where $q = -2$

- Additive inverse, $0 \le a < n$,
  - $b = -a \bmod n$ if $a + b \bmod n = 0$
  - $-a \bmod n \ = n - a$
- Multiplicative inverse
  - $b = a^{-1} \bmod n$ if $ab \bmod n = 1$
  - b exists if and only if $\gcd(a, n) = 1$
- Do modulo anywhere
  - $(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$
  - $(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$
  - $(a \times b) \bmod n = \ [(a \bmod n) \times (b \bmod n)] \bmod n$
  - if $\gcd(b, n) = 1$

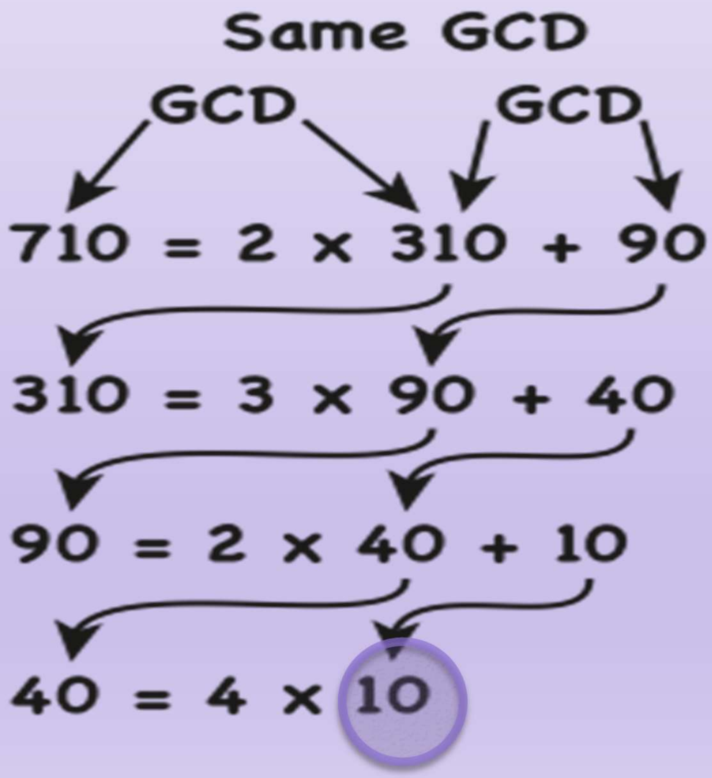
    $$(a/b) \bmod n = \ [(a \bmod n)/(b \bmod n)] \bmod n$$

Congruence

- $a \equiv b \pmod{n}$: a and b are congruent modulo n
  - $a \bmod n = b \bmod n$
- Examples
  - $74 \equiv 28 \pmod{23}$
  - $21 \equiv -9 \pmod{10}$
- $a \equiv b \pmod{n} \Rightarrow n|(a - b)$
- $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n} \Rightarrow a \equiv c \pmod{n}$

Euclidean algorithm

- An $\boldsymbol{efficient}$ algorithm for computing $\mathrm{gcd}(a, b),\ a > b > 0$
- $\mathrm{gcd}(\boldsymbol{a}, \boldsymbol{b}) = \mathrm{gcd}(\boldsymbol{b}, \boldsymbol{a} \bmod \boldsymbol{b})$



Proof:

Example

Dividend	Divisor	Quotient	Remainder
$a = 1160718174$	$b = 316258250$	$q_1 = 3$	$r_1 = 211943424$
$b = 316258250$	$r_1 = 211943434$	$q_2 = 1$	$r_2 = 104314826$
$r_1 = 211943424$	$r_2 = 104314826$	$q_3 = 2$	$r_3 = 3313772$
$r_2 = 104314826$	$r_3 = 3313772$	$q_4 = 31$	$r_4 = 1587894$
$r_3 = 3313772$	$r_4 = 1587894$	$q_5 = 2$	$r_5 = 137984$
$r_4 = 1587894$	$r_5 = 137984$	$q_6 = 11$	$r_6 = 70070$
$r_5 = 137984$	$r_6 = 70070$	$q_7 = 1$	$r_7 = 67914$
$r_6 = 70070$	$r_7 = 67914$	$q_8 = 1$	$r_8 = 2156$
$r_7 = 67914$	$r_8 = 2156$	$q_9 = 31$	$r_9 = 1078$
$r_8 = 2156$	$\boxed{r_9 = 1078}$	$q_{10} = 2$	$r_{10} = 0$

Efficient ➔ the number of divisions $\propto len(a) + len(b)$

Extended Euclidean algorithm

- Given integers a, b and d, find integral solution (x, y) for the equation:

$$\boldsymbol{xa + yb = d}$$

- A solution exists if and only if $\gcd(a, b) | d$
- It suffices to solve $xa + yb = \gcd(a, b)$

- Extended Euclidean algorithm does the work
  - Start with two equations, $a > b > 0$,
    - $1a + 0b = a = r_{-1} \quad - (1)$
    - $0a + 1b = b = r_0 \quad\quad - (2)$
  - $(1) - \lfloor r_{-1}/r_0 \rfloor (2) = r_{-1} - \lfloor \mathrm{r}_{-1}/\mathrm{r}_0 \rfloor r_0 = r_{-1} \bmod r_0 = r_1$

$$\rightarrow x_1 a + y_1 b = r_1 \quad - \quad (3)$$

  - $(2) - \lfloor \mathrm{r}_0/\mathrm{r}_1 \rfloor (3) = r_0 \bmod r_1 = \mathrm{r}_2$

  $\rightarrow x_2 a + y_2 b = r_2 \quad - (4)$

  - Continue till

$$x_n a + y_n b = r_n = \gcd(a, b)$$

Example

- $a = 1759,\ b = 550$
- Equations: $x_i a + y_i b = r_i,\ -1 \le i \le n$
  - $x_{-1} = 1, y_{-1} = 0, r_{-1} = a,\ \ x_0 = 0, y_0 = 1,\ r_0 = b$
  - $q_i = \lfloor r_{i-2} / r_{i-1} \rfloor,\ \ r_i = r_{i-2} \bmod r_{i-1},\ \ i \ge 1$
  - $x_{\mathrm{i}} = x_{i-2} - q_i x_{i-1},\ \ y_i = y_{i-2} - q_i y_{i-1}$

i	r_i	q_i	x_i	Y_i
–1	1759		1	0
0	550		0	1
1	109	3	1	–3
2	5	5	–5	16
3	4	21	106	–339
4	1	1	–111	355
5	0	4		

Prime Numbers

- A prime number has only divisors of 1 and itself
- Prime numbers are central to modern cryptography
- Any integer a > 1 can be factored in a unique way as

$$a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_t^{a_t}$$

where $p_1 < p_2 < \cdots < p_t$ are prime numbers and each a_i is a positive integer

- This is known as the fundamental theorem of arithmetic
- Hereafter, all integers are positive

Primes under 2000

2	101	211	307	401	503	601	701	809	907	1009	1103	1201	1301	1409	1511	1601	1709	1801	1901
3	103	223	311	409	509	607	709	811	911	1013	1109	1213	1303	1423	1523	1607	1721	1811	1907
5	107	227	313	419	521	613	719	821	919	1019	1117	1217	1307	1427	1531	1609	1723	1823	1913
7	109	229	317	421	523	617	727	823	929	1021	1123	1223	1319	1429	1543	1613	1733	1831	1931
11	113	233	331	431	541	619	733	827	937	1031	1129	1229	1321	1433	1549	1619	1741	1847	1933
13	127	239	337	433	547	631	739	829	941	1033	1151	1231	1327	1439	1553	1621	1747	1861	1949
17	131	241	347	439	557	641	743	839	947	1039	1153	1237	1361	1447	1559	1627	1753	1867	1951
19	137	251	349	443	563	643	751	853	953	1049	1163	1249	1367	1451	1567	1637	1759	1871	1973
23	139	257	353	449	569	647	757	857	967	1051	1171	1259	1373	1453	1571	1657	1777	1873	1979
29	149	263	359	457	571	653	761	859	971	1061	1181	1277	1381	1459	1579	1663	1783	1877	1987
31	151	269	367	461	577	659	769	863	977	1063	1187	1279	1399	1471	1583	1667	1787	1879	1993
37	157	271	373	463	587	661	773	877	983	1069	1193	1283		1481	1597	1669	1789	1889	1997
41	163	277	379	467	593	673	787	881	991	1087		1289		1483		1693			1999
43	167	281	383	479	599	677	797	883	997	1091		1291		1487		1697			
47	173	283	389	487		683		887		1093		1297		1489		1699			
53	179	293	397	491		691				1097				1493					
59	181			499										1499					
61	191																		
67	193																		
71	197																		
73	199																		
79																			
83																			
89																			
97																			

Fermat's Little Theorem

- If p is prime, $1 \leq a < p$, then

$$a^{p-1} \bmod p = 1$$

Proof:

- Example
  - $4^{10} \bmod 11 = 1$
  - $6^{28} \bmod 29 = 1$

Euler's Totient Function

$$\phi(n) = p_1^{a_1-1} p_2^{a_2-1} \cdots p_t^{a_t-1} \times (p_1 - 1) \cdots (p_t - 1)$$

$$= |\{a : 1 \le a < n, \gcd(a, n) = 1\}|$$

n	$\phi(n)$
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4

n	$\phi(n)$
11	10
12	4
13	12
14	6
15	8
16	8
17	16
18	6
19	18
20	8

n	$\phi(n)$
21	12
22	10
23	22
24	8
25	20
26	12
27	18
28	12
29	28
30	8

Euler's Theorem

- For $a, n > 0$ and $\gcd(a, n) = 1$,

$$a^{\phi(n)} \bmod n = 1$$

- Fermat’s little theorem is a special case of this theorem
  - $n = p$ is prime, $\phi(p) = p - 1$
- $a^b \bmod n = (a \bmod n)^{b \bmod \phi(n)} \bmod n$
- Example
  - $n = 14, \phi(n) = 6, 5^6 \bmod 14 = 1$
  - $n = 25, \phi(25) = 20, 12^{20} \bmod 25 = 1$

Primality question

- Given an integer $n > 0$, determine whether n is prime
- Brute-force algorithm
  - Find all primes p, $p \leq \sqrt{n}$
  - If any $p|n$, n is not prime; otherwise, n is prime
- Why inefficient?
  - $|\{p \mid p \text{ is prime less than } \sqrt{n}\}| \propto \sqrt{n}/\ln\sqrt{n}$
    - E.g., for 500-digit n, this number is $\geq 10^{249}$
  - Total number of atoms in our universe is about 2^{80}
  - The number of clocks for 1GHz CPU running 1 year is 3.15×10^{16}

- We do have polynomial-time primality algorithm to determine primality of n in time $O(len(n)^{12})$
  - AKS algorithm, by Agrawal, Kayal, and Saxena, 2002
  - Not practical in real applications
- We consider $\boldsymbol{primality\ test}$ algorithm
  - Error with very low probability
    - when n is not prime, we judge it to be prime with very low error probability
    - when n is prime, we judge it to be prime with 100% correct probability
  - These algorithms are practical in real applications, $O(len(n)^3)$

Primality test: theory

- If n is prime,
  - $a^{n-1} \bmod n = 1$ for any $1 \leq a < n$
  - Equation $1 = x^2 \bmod n$ has only two trivial solutions 1 and $n - 1$
- "n is not prime" is confirmed if any one of two conditions does not hold
  - Find $a, 1 \leq a < n$, such that $a^{n-1} \bmod n \neq 1$
  - Find a non-trivial solution for $1 = x^2 \bmod n$

Primality test: practice

- For given n, $n - 1 = 2^r d,$ d is odd
  - Pick random $a, 1 \le a < n,$ compute
    - $x_0 = a^{2^0 d} \bmod n$
    - $x_1 = a^{2^1 d}\ mod\ n$
    - $\cdots$
    - $x_{r-1} = a^{2^{r-1} d} \bmod n$
  - $x_{i+1} = x_i^2 \bmod n, 0 \le i \le r - 2$
  - If any $x_i \ne 1, n - 1$, and $x_i^2\ mod\ n = x_{i+1} = 1,$ then x_i is a non-trivial solution for $1 = x^2 \bmod n$
    - "n is not prime" is confirmed
  - Otherwise, we are not sure yet

- How likely would a random 'a' make us find a non-trivial solution for $1 = x^2 \bmod n$ when n is not prime?
  - By theoretical estimation, ¾ of a's, $1 \leq a < n$, make this happen !!!
- When n is not prime, if we pick t a's, and compute x_i's, the probability that we cannot find a non-trivial solution for the equation is $\leq (1/4)^t$

Primality test: Rabin-Miller algorithm

Input: n: odd

1. Let $n - 1 = 2^r d$, where d is odd
2. For $j = 1$ to t
3. Pick a random number $a, 1 < a < n$

   If $a^{n-1} \bmod n \neq 1$, return (not prime)

   Compute $b_0 = a^d \bmod n, b_1 = a^{2d} \bmod n, \ldots,$

   $b_r = a^{2^r d} \bmod n$

   If any $b_{i-1} \neq 1, n-1, \text{and } b_i = 1$, return (not prime)
4. return (prime)

Primality test: error probability

- $\Pr[\mathrm{RM}(n) = \text{prime} \mid n \text{ is prime}] \; = \; 1$
- $\Pr[\mathrm{RM}(n) = \text{not prime} \mid n \text{ is not prime}] \geq 1 - (1/4)^t$
- For $t = 20$, the error probability is $1/2^{40}$

Primality test: example

- $n = 69, n - 1 = 68 = 2^2 \times 17$
- Pick random $a = 47$ and compute
  - $\mathrm{a}^{68} \bmod 69 = 1$
  - $a^{34} \bmod 69 = 1$
  - $a^{17} \bmod 69 = 47$
- 47 is a solution for $1 = x^2 \bmod 69$
  $\Rightarrow$ 69 is not prime

- $n = 37, n - 1 = 36 = 2^2 \times 9$
  - Pick $a = 4$
    - $a^{36} \bmod 37 = 1$
    - $a^{18} \bmod 37 = 1$
    - $a^{9} \bmod 37 = 36 = -1$
    - $\Rightarrow$ no solutions are found by $a = 4$
  - Pick $a = 20$
    - $a^{36} \bmod 37 = 1$
    - $a^{18} \bmod 37 = 36$
    - $a^{9} \bmod 37 = 31$
    - $\Rightarrow$ no solutions are found by $a = 20$
  - …
- 37 is probably prime

Chinese remainder theorem

- One of the most useful results of number theory
- Find a solution x for the system of linear modulus equations:

$$x \bmod m_i = r_i, 1 \leq i \leq k$$

where $\gcd(m_i, m_j) = 1$ for all $i \neq j$

CRT: solutions

- $M = m_1 m_2 \cdots m_k$
- $M_i = m_1 m_2 \cdots m_{i-1} m_{i+1} \cdots m_k = M/m_i$
  - $\gcd(M_i, m_i) = 1$
- Find C_i such that $C_i M_i \bmod m_i = 1, 1 \le i \le k$
- A solution is

$$x = (r_1 C_1 M_1 + r_2 C_2 M_2 + \cdots + r_k C_k M_k) \bmod M$$

- Why correct?
  - Check $x \bmod m_i = r_i$ for $1 \quad i \le k$
- $x + nM$ is also a solution for any n≥0
- There is a unique solution $x, 0 \le x < M$

CRT: example

- Find a solution x for

$$\begin{cases} x \bmod 3 = 2 \\ x \bmod 7 = 3 \\ x \bmod 8 = 5 \end{cases}$$

- $M_1 = 56, M_2 = 24, M_3 = 21, M = 168$
- $C_1 = 2, C_2 = 5, \; C_3 = 5$
- $x = (2 \times 2 \times 56 + 3 \times 5 \times 24 + 5 \times 5 \times 21) \bmod 168 = 101$

CRT: computing inverse

- How to find C_i such that $C_i M_i \bmod m_i = 1$?
- Use extended Euclidean algorithm to solve (x, y) for

$$xM_i + ym_i = \gcd(M_i, m_i) = 1$$

- $C_i = x \bmod m_i$ since $xM_i \bmod m_i = (1 - ym_i) \bmod m_i = 1$

CRT: insight

- Number system mapping
  - $M = m_1 m_2 \cdots m_k,\ \mathrm{gcd}(m_i, m_j) = 1 \text{ for } i \neq j$
  - N1: $x,\ 0 \leq x \leq M - 1$
    - $\mathrm{mod}\ M$
  - N2: $(r_1, r_2, \ldots, r_k), 0 \leq r_i \leq m_i - 1$
    - $(\mathrm{mod}\ m_1, \mathrm{mod}\ m_2, \ldots, \mathrm{mod}\ m_k)$
- One-to-one mapping between N1 and N2
  - N1→N2: $x \rightarrow (x \bmod m_1, x \bmod m_2, \ldots, x \bmod m_k)$
  - N2→N1: $(r_1, r_2, \ldots, r_k) \rightarrow x,$ by CRT
- Operations on N2 are more efficient than operations on N1 since the numbers are smaller

CRT: mapping example

- $M = 1813 = 37 \times 49 = m_1 m_2$
- N1: $x, 0 \leq x \leq 1812$
- N2: $(r_1, r_2), 0 \leq r_1 \leq 36, 0 \leq r_2 \leq 48$
- Example: $678 \leftrightarrow (12, 41), 973 \leftrightarrow (11,42)$
- Addition
  - $678 + 973 \bmod 1813 = 1651$
  - $(12, 41) + (11, 42) = (23, 34) \rightarrow 1651$
- Multiplication
  - $678 \times 973 \bmod 1813 = 1575$
  - $(12, 41) \times (11, 42) = (12 \times 11 \bmod 37, 41 \times 42 \bmod 49) = (21, 7) \rightarrow 1575$

Power of integers modulo n

- For $a, n > 0$, the ***order*** of $a \pmod{n}$ is

$$ord_n(a) = \min_{m>0} a^m \bmod n = 1$$

- If $\gcd(a, n) = 1$, m exists and $m|\phi(n)$
- If $ord_n(a) = \phi(\text{n})$, a is called a $\boldsymbol{primitive\ root\ of\ n}$ (or $\boldsymbol{primitive\ root\ modulo\ n}$)
  - $a^1, a^2, \ldots, a^{\phi(n)}$ are all distinct and relatively prime to n
- Only $n = 2, 4, p^\alpha, 2p^\alpha$ have primitive roots, where p is odd prime and $\alpha > 0$
- A primitive root of n is a generator of Z_n^*, which generates the whole Z_n^* by taking powers, i.e., $Z_n^* = \{a^i | i \geq 0\}$

- Powers of integers modulo 19, $\phi(19) = 18$
- $p = 19$ has 6 primitive roots 2, 3, 10, 13, 14 and 15

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Discrete logarithm: mod p

- Operations are on multiplicative group Z_p^* of $p-1$ elements
- For prime p and g, primitive root of p, for $1 \leq y < p$,

$$\mathrm{dlog_{g,p}}(y) = x, \text{where } y = g^x \bmod p$$

- $0 \leq \mathrm{dlog_{g,p}}(y) \leq p-2$
- $\mathrm{dlog_{g,p}}(1) = 0$
- $\mathrm{dlog_{g,p}}(g) = 1$
- $\mathrm{dlog_{g,p}}(y_1 y_2) = [\mathrm{dlog_{g,p}}(y_1) + \mathrm{dlog_{g,p}}(y_2)] \bmod (p-1)$
  - If working on group Z_n^*, should be "mod $\phi(n)$"

(a) Discrete logarithms to the base 2, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

0

(b) Discrete logarithms to the base 3, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) Discrete logarithms to the base 10, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) Discrete logarithms to the base 13, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) Discrete logarithms to the base 14, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

Discrete logarithm: example

- $p = 1999, a = 47$
- What are $\mathrm{dlog}_{47,1999}867$ and $\mathrm{dlog}_{47,1999}942$?
  - By brute-fore search $x, 1 \leq x \leq p - 2$, we got $x = 305$ and 1853, respectively
- What algorithms compute discrete logarithm more effective than brute-force search?

Baby-step-giant-step algorithm

- What is $x = \mathrm{dlog}_{47,1999}(866)$?
- Method
  - Let $x = im + j, m = \lceil\sqrt{p}\rceil, 1 \leq i, j \leq m$
  - $g^{im+j} \equiv y \pmod{p} \Rightarrow g^j \bmod p = y(g^{-m})^i \bmod p$
  - Compute $b_j = g^j \bmod p, a_i = y(g^{-m})^i \bmod p$, for $1 \leq i, j \leq m$
  - Find (i_0, j_0) such that $a_{i_0} = b_{j_0}$. Then $x = i_0 m + j_0$
- $m = \lceil\sqrt{1999}\rceil = 45, g^{-m} \bmod p = 167$
- $(i_0, j_0) = (16, 19) \Rightarrow a_{i_0} = b_{j_0} = 1232,$

$$\Rightarrow x = i_0 m + j_0 = 739$$

- Time complexity is $O(\sqrt{p})$. Effective?

Discrete logarithm: calculation

- Modular exponentiation
  - given x, g and p, compute $y = g^x \bmod p$
  - y can be computed by square-and-multiply algorithm efficiently, $\propto len(p)^3$
- Discrete logarithm
  - given y, g and p, compute $x = \mathrm{dlog}_{g,p}(y)$
  - Best algorithm takes time

$$e^{(1.923+o(1))\cdot(\ln\, p)^{\frac{1}{3}}(\ln(\ln p))^{\frac{2}{3}}}$$

- It is infeasible theoretically since the complexity is not poly-time of $len(p)$
- Nevertheless, we can still compute for large p practically
  - take $1.923 + \mathrm{o}(1) = 1.95$
  - p is 512-bit long, this value is 3.1×10^{19}
  - p is 1024-bit long, this value is 1.6×10^{26}
  - p is 2048-bit long, this value is 4.8×10^{35}