

# MA7010 – Number Theory for Cryptography - Assignment 2

Ajeesh Thattukunnel Vijayan

January 11<sup>th</sup> 2024

## 1 Answers

1. Lower Range = 600, Upper Range = 750. Consider all the numbers  $n$  in your range. Divide the set into two subsets: A - the subset consisting of all  $n$  where there is at least one primitive root modulo  $n$ ; B – the subset consisting of all  $n$  where no primitive roots exist modulo  $n$

**Answer:** The below json snippets show the two sets:

```
1      {
2          "Numbers With Primitive Roots (A)": [
3              "601", "607", "613", "614", "617", "619", "622", "625",
4              "626", "631", "634", "641", "643", "647", "653", "659",
5              "661", "662", "673", "674", "677", "683", "686", "691",
6              "694", "698", "701", "706", "709", "718", "719", "722",
7              "727", "729", "733", "734", "739", "743", "746"
8          ]
9      }
10
11     {
12         "Numbers Without Primitive Roots (B)": [
13             "600", "602", "603", "604", "605", "606", "608", "609",
14             "610", "611", "612", "615", "616", "618", "620", "621",
15             "623", "624", "627", "628", "629", "630", "632", "633",
16             "635", "636", "637", "638", "639", "640", "642", "644",
17             "645", "646", "648", "649", "650", "651", "652", "654",
18             "655", "656", "657", "658", "660", "663", "664", "665",
19             "666", "667", "668", "669", "670", "671", "672", "675",
20             "676", "678", "679", "680", "681", "682", "684", "685",
21             "687", "688", "689", "690", "692", "693", "695", "696",
22             "697", "699", "700", "702", "703", "704", "705", "707",
23             "708", "710", "711", "712", "713", "714", "715", "716",
24             "717", "720", "721", "723", "724", "725", "726", "728",
25             "730", "731", "732", "735", "736", "737", "738", "740",
26             "741", "742", "744", "745", "747", "748", "749", "750"
27         ]
28     }
29
```

Listing 1: List of Numbers With and Without Primitive Roots

The Rust code for generating the above result is below:

```
1
2      ///
3      /// Returns a vec of primitive roots for the integer
4      ///
```

```

5      /// # Arguments
6      /// * n: BigInt
7      ///
8      /// Steps:
9      /// This function uses trial and error to find primitive roots
10     /// associated to an Integer
11     ///
12     /// 1. Find all coprime numbers less than 'n'
13     ///     (coprime_nums_less_than_n)
14     /// 2.  $\phi(n)$  = total number of coprimes
15     /// 3. Find all the divisors of  $\phi(n)$ . Order of an
16     ///     element in the Modulo n group will be equal to
17     ///     any of the divisor values.
18     /// 4. Find the order of each of the coprimes to n one by one
19     ///     (skip 1 from the list of
20     ///     coprimes as 1 is a trivial root) ('use utils::modular_pow)
21     /// 5. if order of a coprime integer equals  $\phi(n)$ , that coprime
22     ///     is a primitive root
23     ///
24     /// The above steps are executed against all coprimes to n and
25     /// returns an integer vector with primitive roots
26     ///
27     pub fn primitive_roots_trial_n_error(n: &BigInt) -> Vec<BigInt> {
28         let mut primitive_roots: Vec<BigInt> = Vec::new();
29         let mut has_primitive_roots: bool = false;
30
31         let nums_coprime_n: Vec<BigInt> = coprime_nums_less_than_n(n);
32         let phi_n = BigInt::from(nums_coprime_n.len());
33         //
34         let divisors_phi_n = divisors_of_n(&phi_n);
35
36         for a in nums_coprime_n {
37             let mut has_order_phi: bool = true;
38             for order in divisors_phi_n.iter() {
39                 if modular_pow(&a, order, n) == BigInt::one() {
40                     if *order != phi_n {
41                         has_order_phi = false;
42                     }
43                 }
44             }
45
46             if has_order_phi {
47                 primitive_roots.push(a);
48                 has_primitive_roots = true;
49                 break;
50             }
51         }
52
53         if has_primitive_roots {
54             let orders_coprime_phi_n: Vec<BigInt> =
55             coprime_nums_less_than_n(&phi_n);
56             // first coprime number is 1 and we are skipping that
57             // when calculating power
58             for order in orders_coprime_phi_n.iter().skip(1) {
59                 primitive_roots.push(modular_pow(&primitive_roots[0], order,
60 n));
61             }
62
63             primitive_roots.sort();
64
65             for (i, num) in primitive_roots.clone().iter().enumerate() {

```

```

65         if num == &BigInt::one() {
66             primitive_roots.remove(i);
67             continue;
68         }
69
70         if modular_pow(num, &phi_n, n) != BigInt::one() {
71             primitive_roots.remove(i);
72         }
73     }
74
75     primitive_roots
76 }
77
78 ///
79 /// Generates a list of integers less than n and co-prime to n.
80 ///
81 pub fn coprime_nums_less_than_n(n: &BigInt) -> Vec<BigInt> {
82     let mut coprimes: Vec<BigInt> = Vec::new();
83     let r = range(BigInt::from(1u64), n.clone());
84
85     for num in r {
86         if n.gcd_euclid(&num) == BigInt::one() {
87             coprimes.push(num)
88         }
89     }
90     coprimes.sort();
91     coprimes
92 }
93
94 ///
95 /// Get list of divisors of a number n > 2
96 ///
97 pub fn divisors_of_n(n: &BigInt) -> Vec<BigInt> {
98     let mut divisors: Vec<BigInt> = Vec::new();
99     let mut primes = vec![BigInt::from(2u64)];
100    let p_factors_n = n.prime_factors(&mut primes);
101    let p_factors_n = p_factors_n
102        .iter()
103        .map(|(p, _)| p.clone())
104        .collect::<Vec<BigInt>>();
105
106    for p in p_factors_n {
107        let mut i = 0;
108        loop {
109            let pow = p.pow(i);
110            if n % &pow == BigInt::zero() {
111                divisors.push(n / &pow);
112                divisors.push(pow);
113                i += 1;
114            } else {
115                break;
116            }
117        }
118    }
119    divisors.sort();
120    divisors.dedup();
121    divisors
122 }
123

```

Listing 2: Primitive Roots Calculation

2. a. Explain why we can always find a primitive root modulo  $p$  when  $p$  is a prime.

**Answer:** When  $n = 2$ , 1 is always a primitive root as  $1^{\phi(2)} \equiv 1 \pmod{2}$

By Fermat's Little Theorem,  $a^{p-1} \equiv 1 \pmod{p}$  where  $p$  is a prime. This means, there exists an  $a \in \mathbb{Z}/p\mathbb{Z}$  satisfying this theorem. And for a prime,  $p$ ,

$$\phi(p) = p - 1 \implies a^{p-1} = a^{\phi(p)} \equiv 1 \pmod{p}$$

- b. Express the number of primitive roots that exist modulo  $p$  using the Euler Totient function and show that your answer correctly predicts the number of primitive roots for all primes in your given range.

**Answer:** The number of primitive roots associated with an integer  $n$  is given by  $\phi(\phi(n))$

When  $n$  is a prime, namely  $p$ ,  $\phi(\phi(p)) = \phi(p - 1)$ . The below table verifies this value against the number calculated using trial and error for all primes in the range  $600 \leq p \leq 750$ .

Prime	Primitive Roots Count - Trial and Error	$\phi(p - 1)$
601	160	160
607	200	200
613	192	192
617	240	240
619	204	204
631	144	144
641	256	256
643	212	212
647	288	288
653	324	324
659	276	276
661	160	160
673	192	192
677	312	312
683	300	300
691	176	176
701	240	240
709	232	232
719	358	358
727	220	220
733	240	240
739	240	240
743	312	312

Table 1: Primitive Roots Count

The Rust code for generating the above result is below. It calls the function listed in code 2.

```

1
2           PrimitiveRootsCommands::Ass2Question2b(r) => {
3   let start = r.start;
4   let end = r.end;
5
6   let mut result: Vec<HashMap<String, String>> = Vec::new();
7   let (primes_in_range, _) =
8   find_primes_in_range_trial_division_parallel(start, end);
9   for p in primes_in_range.iter() {
10      let primitive_roots = primitive_roots_trial_n_error(p);
11      let phi_phi_n = euler_totient_phi(&(p - BigInt::one()));
12      let mut item: HashMap<String, String> = HashMap::new();
13      item.insert("Prime".to_string(), p.to_string());

```

```

13         item.insert("Euler_Totient(p-1)".to_string(), phi_phi_n.
14         to_string());
15         item.insert(
16         "Prim Roots Count - Trial and Error".to_string(),
17         primitive_roots.len().to_string(),
18         );
19         result.push(item);
20     }
21     println!("{}", serde_json::to_string_pretty(&result).unwrap
22     ());

```

Listing 3: Primitive Roots - Euler's Totient Function Verification

We can use the below command to see the above result:

```

1  .\target\release\nt-assignments.exe primitive-roots ass2-question2b -s 600 -e 750
2
3

```

Listing 4: Verify Primitive Roots Counting using Totient Function

- c. For the same range as Question 1 use the command ifactors in Maple to find the set  $C$  whose elements consist of numbers of the form  $p^k$  ( $p > 2, k \geq 1$ ) or  $2p^k$  ( $p > 2, k \geq 1$ )

Number	Form	Number	Form
601	$601^1$	674	$2^1 \times 337^1$
607	$607^1$	677	$677^1$
613	$613^1$	683	$683^1$
614	$2^1 \times 307^1$	686	$2^1 \times 7^3$
617	$617^1$	691	$691^1$
619	$619^1$	694	$2^1 \times 347^1$
622	$2^1 \times 311^1$	698	$2^1 \times 349^1$
625	$5^4$	701	$701^1$
626	$2^1 \times 313^1$	706	$2^1 \times 353^1$
631	$631^1$	709	$709^1$
634	$2^1 \times 317^1$	718	$2^1 \times 359^1$
641	$641^1$	719	$719^1$
643	$643^1$	722	$2^1 \times 19^2$
647	$647^1$	727	$727^1$
653	$653^1$	729	$3^6$
659	$659^1$	733	$733^1$
661	$661^1$	734	$2^1 \times 367^1$
662	$2^1 \times 331^1$	739	$739^1$
673	$673^1$	743	$743^1$
746	$2^1 \times 373^1$	-	-

Table 2: Numbers of the form  $p^k, 2p^k$ 

The Rust code for generating the above result is below.

```

1
2     /// It checks the existence of primitive roots modulo n
3     /// and returns the number of primitive roots
4     pub fn is_integer_of_form_pk_2pk(n: &BigInt) -> Vec<(BigInt,
5     usize)> {
6         let (zero, two) = (BigInt::zero(), BigInt::from(2u64));
7         let mut primes = vec![BigInt::from(2u64)];

```

```

7      let p_factors = n.prime_factors(&mut primes);
8      if p_factors.len() < 1 || p_factors.len() > 2 {
9          return vec![];
10     }
11
12     // p_factors is a sorted vector
13     // check the criteria on on a clone of p_factors
14     let mut p_factors_clone = p_factors.clone();
15     match p_factors_clone.len() {
16         // If the prime factors have a length of 1,
17         // then it must be of the form p^k. Hence, if
18         // p = 2, fail
19         1 => {
20             if let Some(first) = p_factors_clone.pop() {
21                 if &first.0 == &two {
22                     return vec![];
23                 }
24             }
25         }
26         // If the prime factors have a length of 2,
27         // then the first factor is 2^1 and the second factor is
28         p^k
29         2 => {
30             let first = p_factors_clone.remove(0);
31             if &first.0 == &two {
32                 // since the p_factors vec is prepared in sorted
33                 form and without
34                 // duplicates, we only need to check the first
35                 if first.1 > 1 {
36                     return vec![];
37                 }
38             } else {
39                 return vec![];
40             }
41         }
42         _ => return vec![],
43     }
44     p_factors
45 }

```

Listing 5: Numbers of the form  $p^k$ ,  $2p^k$ 

d. Hence form a conjecture about when primitive roots do and don't exist

**Answer:** From 2(c), it's evident that numbers of the form  $p^k$  and  $2p^k$  have primitive roots. A special case is when  $p = 2$  and  $p = 4$ . When  $p = 2$ , 1 is a primitive root. When  $p = 4$ ,  $\phi(4) = 2$  and  $3^{\phi(4)} \equiv 1 \pmod{4}$ . Hence we can claim that if a number is of the form  $2, 4, p^k$ , or  $2p^k$  where  $p > 2$  and  $k \geq 1$ , that prime has primitive roots.

3. Suppose  $n$  has the form  $n = pq$  where  $p$  and  $q$  are different primes both  $> 2$ .

(a) What is  $\phi(n)$  in terms of  $p$  and  $q$ ?

**Answer:**  $\phi(n) = \phi(p.q) = \phi(p) \cdot \phi(q) = (p-1) \cdot (q-1)$

(b) Suppose  $a$  is relatively prime to  $pq$ . Explain why

i.  $a^{p-1} \equiv 1 \pmod{p}$

**Answer:** Given  $p$  and  $q$  are distinct primes. Since  $(a, pq) = 1$ ,  $a$  is relatively prime to both  $p$  and  $q$ . Hence by Fermat's Little Theorem,  $a^{p-1} \equiv 1 \pmod{p}$ .

ii.  $a^{q-1} \equiv 1 \pmod{q}$

**Answer:** Given  $p$  and  $q$  are distinct primes. Since  $(a, pq) = 1$ ,  $a$  is relatively prime to both  $p$  and  $q$ . Hence by Fermat's Little Theorem,  $a^{q-1} \equiv 1 \pmod{q}$ .

iii.  $m = \text{lcm}(p-1, q-1)$  is less than  $(p-1)(q-1)$

**Answer:** Since both  $p$  and  $q$  are odd primes,  $p-1$  and  $q-1$  are even. Let  $p-1 = 2j$  and  $q-1 = 2k$ . Then  $(p-1, q-1) = (2j, 2k) = 2(j, k)$ . We can see that there will be a factor of 2 at a minimum when number are even. LCM is given by

$\text{lcm}(p-1, q-1) = \frac{(p-1)(q-1)}{\gcd(p-1, q-1)}$ , which means  $m = \text{lcm}(p-1, q-1)$  equals  $(p-1)(q-1)$

only when  $\gcd(p-1, q-1) = 1$ . But here we have  $\gcd > 1$  and hence

$m = \text{lcm}(p-1, q-1) < (p-1)(q-1)$

iv.  $a^m \equiv 1 \pmod{(p-1)(q-1)}$

**Answer:**

(c) Hence explain why numbers of the form  $n$  have no primitive roots.

**Answer:** Suppose  $n = p.q$ , where  $p$  and  $q$  are primes has primitive roots. This means there exists  $a \in (\mathbb{Z}/pq\mathbb{Z})^\times$  such that  $\text{ord}_{pq}(a)$  will be  $m = \phi(n) = \phi(p.q) = (p-1).(q-1)$ . Also  $\gcd(a, pq) = 1$ .

Also,

$$a^m \equiv 1 \pmod{pq} \quad (1)$$

$$\iff a^m \equiv 1 \pmod{p}, a^m \equiv 1 \pmod{q} \text{ (By Chinese Remainder Theorem)} \quad (2)$$

$$\iff m \equiv 0 \pmod{p-1}, m \equiv 0 \pmod{q-1} \quad (3)$$

(Because by Fermat's Little Theorem,  $a^{p-1} \equiv 1 \pmod{p}$  and  $a^{q-1} \equiv 1 \pmod{q}$ )

$$\iff (p-1)|m, (q-1)|m \quad (4)$$

$$\iff \text{lcm}(p-1, q-1)|m \quad (5)$$

This means that  $\text{ord}_p(a) = \text{lcm}(p-1, q-1) < (p-1)(q-1)$  as we have seen in 3(b)iii and it's a contradiction from our initial assumption that  $n = p.q$  has primitive roots.

(d) Show that all numbers of the form  $n = pq$  ( $p$  and  $q$  both odd primes) in your range are included in set B.

**Answer:**

```

1      {
2          "Numbers Without Primitive Roots (B)": [
3              "600", "602", "603", "604", "605", "606", "608", "609",
4              "610", "611", "612", "615", "616", "618", "620", "621",
5              "623", "624", "627", "628", "629", "630", "632", "633",
6              "635", "636", "637", "638", "639", "640", "642", "644",
7              "645", "646", "648", "649", "650", "651", "652", "654",
8              "655", "656", "657", "658", "660", "663", "664", "665",
9              "666", "667", "668", "669", "670", "671", "672", "675",
10             "676", "678", "679", "680", "681", "682", "684", "685",
11             "687", "688", "689", "690", "692", "693", "695", "696",
12             "697", "699", "700", "702", "703", "704", "705", "707",
13             "708", "710", "711", "712", "713", "714", "715", "716",
14             "717", "720", "721", "723", "724", "725", "726", "728",
15             "730", "731", "732", "735", "736", "737", "738", "740",
16             "741", "742", "744", "745", "747", "748", "749", "750"
17         ]
18     }
19 
```

Listing 6: List of Numbers Without Primitive Roots

And the below table shows the list of numbers of the form  $p.q$  in the range  $600 \leq n \leq 750$ , which is a subset of the set  $B$  above 6.

Number	Form	Number	Form
611	$13^1 \times 47^1$	687	$3^1 \times 229^1$
623	$7^1 \times 89^1$	689	$13^1 \times 53^1$
629	$17^1 \times 37^1$	695	$5^1 \times 139^1$
633	$3^1 \times 211^1$	697	$17^1 \times 41^1$
635	$5^1 \times 127^1$	699	$3^1 \times 233^1$
649	$11^1 \times 59^1$	703	$19^1 \times 37^1$
655	$5^1 \times 131^1$	707	$7^1 \times 101^1$
667	$23^1 \times 29^1$	713	$23^1 \times 31^1$
669	$3^1 \times 223^1$	717	$3^1 \times 239^1$
671	$11^1 \times 61^1$	721	$7^1 \times 103^1$
679	$7^1 \times 97^1$	723	$3^1 \times 241^1$
681	$3^1 \times 227^1$	731	$17^1 \times 43^1$
685	$5^1 \times 137^1$	737	$11^1 \times 67^1$
-	-	745	$5^1 \times 149^1$
-	-	749	$7^1 \times 107^1$

Table 3: Numbers of the form  $p.q$

4. Use the BabyStepsGiantSteps algorithm to find discrete logarithms  $x$  of  $b \bmod n$  for the primitive root  $a$  for each of the two examples assigned to you in the table below. Verify that your answer is correct by calculating  $a^x \bmod m$  by hand using the method of modular exponentiation.

*Note: Somehow I couldn't make it work the Baby Steps Giant Steps Algorithm as we learned in the class. I checked the Wikipedia and it's the same as in the class. I do not know where did it go wrong. I was getting a smaller value that expected. So I followed the steps from some Youtube videos([Video1](#), [Video2](#)). The steps are similar with some minor variations in the values we calculate. Hope that's fine.*

- (a) **Answer:** Given  $a = 21, b = 47, n = 71$ . We want to solve for  $t$  in the congruence:  
 $21^t \equiv 47 \pmod{71}$

We have  $\phi(71) = 70$ .

Step 1. Set  $m = \lceil \sqrt{71} \rceil = 9$

Step 2. Calculating  $a^{mj} \pmod{71}; 0 \leq j < m$

j	$a^{mj} \pmod{71}$	j	$a^{mj} \pmod{71}$	j	$a^{mj} \pmod{71}$
0	$21^{9 \cdot 0} = 1$	3	$21^{9 \cdot 3} = 35$	6	$21^{9 \cdot 6} = 18$
1	$21^{9 \cdot 1} = 42$	4	$21^{9 \cdot 4} = 50$	7	$21^{9 \cdot 7} = 46$
2	$21^{9 \cdot 2} = 60$	5	$21^{9 \cdot 5} = 41$	8	$21^{9 \cdot 8} = 15$

Step 3. Solve for  $b.a^{-i}; 0 \leq i < m$

i	$b.a^{-i} \pmod{71}$	i	$b.a^{-i} \pmod{71}$
0	$47.21^0 = 47$	4	$47.21^{-4} = 47.21^{66} = 69$
1	$47.21^{-1} = 47.21^{69} = 9$	5	$47.21^{-5} = 47.21^{65} = 54$
2	$47.21^{-2} = 47.21^{68} = 41$	6	$47.21^{-6} = 47.21^{64} = 33$
3	$47.21^{-3} = 47.21^{67} = 29$	7	$47.21^{-7} = 47.21^{63} = 32$
-	-	8	$47.21^{-8} = 47.21^{62} = 59$



Step 4. We found a collision in both the tables for  $value = 41$  where  $i = 2$  and  $mj = 9 \times 5$

Step 5. We calculate  $t = (mj + i) \pmod{71} = (9 \times 5 + 2) \pmod{71} \equiv 47 \pmod{71}$

$$\implies 21^{47} \equiv 47 \pmod{71}$$

Step 6. Verifying the answer using Fast Modular Exponentiation:

$$21^2 \equiv 15 \pmod{71}$$

$$\therefore 21^4 = (21^2)^2 = 15^2 \pmod{71} \equiv 12 \pmod{71}$$

$$\implies 21^8 = (21^4)^2 = 12^2 \equiv 2 \pmod{71}$$

$$\implies 21^{16} = (21^8)^2 = 2^2 \equiv 4 \pmod{71}$$

$$\implies 21^{32} = (21^{16})^2 = 4^2 \equiv 16 \pmod{71}$$

$$\implies 21^{40} = 21^{32} \times 21^8 \pmod{71} \equiv 16 \times 2 \pmod{71} \equiv 32 \pmod{71}$$

We will calculate now  $21^7 \pmod{71}$  using the below steps:

The binary representation for  $7 = [111] \sim [d_2 d_1 d_0]$

Let  $a = 1$  and  $s = 21$

$k = 0$  : Since  $d_k = 1, a = a \times s = 21 \pmod{71}, s = s^2 = 15 \pmod{71}$

$k = 1$  : Since  $d_k = 1, a = a \times s = 31 \pmod{71}, s = s^2 = 12 \pmod{71}$

$k = 2$  : Since  $d_k = 1, a = a \times s = 17 \pmod{71},$

$$\implies 21^7 \equiv 17 \pmod{71}$$

$$\therefore 21^{47} = 21^{40} \times 21^7 = 32 \times 17 \equiv 47 \pmod{71} \text{ and hence the answer}$$

(b) **Answer:** Given  $a = 26, b = 24, n = 53$ . We want to solve for  $t$  in the congruence:

$$26^t \equiv 24 \pmod{53}$$

Step 1. Set  $m = \lceil \sqrt{53} \rceil = 8$

Step 2. –

Step 3. Calculating  $a^{mj} \pmod{53}; 0 \leq j < m$  & Solve for  $b.a^{-i} \pmod{53}; 0 \leq i < m$  (Step 2 and 3 tables below side-by-side)

$$26^{-1} \equiv 27 \pmod{53}$$

j	$a^{mj} \pmod{71}$
0	$26^{8 \cdot 0} \equiv 1$
1	$26^{8 \cdot 1} \equiv 47$
2	$26^{8 \cdot 2} \equiv 36$
3	$26^{8 \cdot 3} \equiv 49$
4	$26^{8 \cdot 4} \equiv 24$
5	$26^{8 \cdot 5} \equiv 15$
6	$26^{8 \cdot 6} \equiv 16$
7	$26^{8 \cdot 7} \equiv 10$

Table 4: Step 2

i	$b.a^{-i} \pmod{53}$
0	$24.26^0 = 24.26^{52} = 26$
1	$24.26^{-1} = 24.26^{51} = 5$
2	$24.26^{-2} = 24.26^{50} = 43$
3	$24.26^{-3} = 24.26^{49} = 20$
4	$24.26^{-4} = 24.26^{48} = 13$
5	$24.26^{-5} = 24.26^{47} = 27$
6	$24.26^{-6} = 24.26^{46} = 52$
7	$24.26^{-7} = 24.26^{45} = 2$
8	$24.26^{-8} = 24.26^{44} = 49$

Table 5: Step 3

Step 4. We found a collision in both the tables for  $value = 49$  where  $i = 8$  and  $mj = 8 \times 3$

Step 5. We calculate  $t = (mj + i) \pmod{53} = (8 \times 3 + 8) \pmod{53} \equiv 32 \pmod{53}$

$$\implies 26^{32} \equiv 24 \pmod{53}$$

Step 6. Verifying the answer using Fast Modular Exponentiation:

$$\begin{aligned}
 26^2 &\equiv 40 \pmod{53} \\
 \therefore 26^4 &= (26^2)^2 = 40^2 \equiv 10 \pmod{53} \\
 \implies 26^{16} &= (26^4)^4 = 10^4 \equiv 36 \pmod{53} \\
 \implies 26^{32} &= (26^{16})^2 = 36^2 \equiv 24 \pmod{53} \text{ and hence the answer}
 \end{aligned}$$

5. Use the Pohlig Hellmann algorithm to find in the cyclic group of order  $n$  with the generating element  $a$  for both the examples assigned to you below. Verify your answer in Maple.

(a) **Answer:**  $a = x^{11}, b = x^{41}, n = 343$

$$n = 343 = 7^3$$

We will write  $G$  as  $G = \{x^i | 0 \leq i \leq 342, x^{342} = 1\}$ . Also  $x^{11}$  generates  $G$  as  $(11, 343) = 1$ . So we set  $p = 7, e = 3, g = x^{11}, h = x^{41}$

Step 1. Setting  $x_0 = 0$  and let  $n = p^e, p^{e-1} = p^2 = 49$

When  $k = 0$ :

$$s = g^{p^{e-1}} = g^{n/p} = (x^{11})^{49} = x^{539} = x^{196}$$

$$h_0 = (g^{-x_0} \times h)^{p^{e-1}} = h^{49} = (x^{41})^{49} = x^{294}$$

Since  $p = 7$ , test for  $d_0 \in \{0, 1, 2, 3, 4, 5, 6\}$  satisfying  $s^{d_0} = h_0$

$\therefore$ , for  $d_0 = 5$ , we have  $s^{d_0} = h_0$ , so  $d_0 = 5$

$$x_1 = x_0 + p^0 \cdot d_0 = 0 + 1 \cdot 5 = 5$$

Step 2. When  $k = 1$  we have  $x_1 = 5, p^{e-2} = p^1 = 7$

$$h_1 = (g^{-x_1} \times h)^{p^{e-2}} = (g^{-5} \times h)^7 = (x^{-55} \times x^{41})^7 = x^{-98} = x^{245}$$

Searching for  $d_1 \in \{0, 1, 2, 3, 4, 5, 6\}$  satisfying  $s^{d_1} = h_1$

$\therefore$ ,  $r = 3$  satisfies the condition. So  $d_1 = 3$

$$x_2 = x_1 + p^1 \cdot d_1 = 5 + 7 \times 3 = 26$$

Step 3. When  $k = 2$ , we have  $x_2 = 26, p^{e-3} = 1$

$$h_2 = (g^{-x_2} \times h)^{p^{e-3}} = (g^{-26} \times h)^1 = x^{-286} \times x^{41} = x^{-245} = x^{98}$$

Searching for  $d_2$ , we get  $d_2 = 4$

$$x_3 = x_2 + p^2 \cdot d_2 = 26 + 49 \times 4 = 222$$

$x = 222$  is the logarithm we wanted.

Below is the Maple Verification result:

$$G = \{x^i | 0 \leq i \leq 342, x^{342} = 1\}$$

$$\gcd(11, 343) = 1, x^{11} \text{ generates the Group.}$$

$$p := 7; e := 3; g := x^{11}; h := x^{41};$$

$$p := 7$$

$$e := 3$$

$$g := x^{11}$$

$$h := x^{41}$$

(6)

Step1:

$$x0 := 0;$$

$$x0 := 0$$

(7)

$$s := x^{11 \cdot 49 \bmod 343}; h0 := x^{41 \cdot 49 \bmod 343};$$

$$s := x^{196}$$

$$h0 := x^{294} \quad (8)$$

*Searching for*  $d0$ ;  $d0 = 5$  *satisfies*  $^{d0} = h0$   
 $d0 := 5$ ;

$$d0 := 5 \quad (9)$$

$$x^{196 \cdot 5 \mod 343};$$

$$x^{294} \quad (10)$$

$$x1 := x0 + p^0 \cdot d0;$$

$$x1 := 5 \quad (11)$$

Step 2:

$$h1 := x^{(-55+41) \cdot 7 \mod 343};$$

$$h1 := x^{245} \quad (12)$$

*Searching for*  $d1$ ;  $d1 = 3$  *satisfies*  $^{d1} = h1$   
 $d1 := 3$ ;

$$d1 := 3 \quad (13)$$

$$x^{196 \cdot 3 \mod 343};$$

$$x^{245} \quad (14)$$

$$x2 := x1 + p^1 \cdot d1;$$

$$x2 := 26 \quad (15)$$

Step 3:

$$h2 := x^{-286+41 \mod 343};$$

$$h2 := x^{98} \quad (16)$$

*Searching for*  $d2$ ;  $d2 = 4$  *satisfies*  $^{d2} = h2$   
 $d2 := 4$ ;

$$d2 := 4 \quad (17)$$

$$x^{196 \cdot 4 \mod 343};$$

$$x^{98} \quad (18)$$

$$x3 := x2 + p^2 \cdot d2;$$

$$x3 := 222 \quad (19)$$

$x3$  is our logarithm

(b) **Answer:**  $a = x^{13}, b = x^{157}, n = 3267$

$$n = 3267 = 3^3 \times 11^2 = 27 \times 121$$

We will write  $G$  as  $G = \{x^i | 0 \leq i \leq 3266, x^{342} = 1\}$ . Also  $x^{13}$  generates  $G$  as  $(13, 3267) = 1$ .  
Step 1.

$$\begin{aligned} g1 &= g^{121} = x^{13 \times 121} \pmod{27} = x^7 \\ h1 &= h^{121} = x^{157 \times 121} \pmod{27} = x^{16} \end{aligned}$$

We will need to find the logarithm of  $h1 = x^{16}$  in the cyclic group of order 27 generated by  $g1 = x^7$ . With some trial and error, we get  $\log x^{16} = 10$ , i.e.,  $x^{7 \pmod{27}} = x^{16}$ . Hence we get the below congruence:

$$x \equiv 10 \pmod{27} \quad (20)$$

Step 2.

$$\begin{aligned} g2 &= g^{27} = x^{13 \times 27} \pmod{121} = x^{109} \\ h2 &= h^{27} = x^{157 \times 27} \pmod{121} = x^4 \end{aligned}$$

Let's find the logarithm of  $h2 = x^4$  in the cyclic group of order 121 generated by  $g2 = x^{109}$ . With some trial and error, we get  $\log x^4 = 40$ , i.e.,  $x^{109 \times 40} \pmod{121} = 4$ . We get the following congruence:

$$x \equiv 40 \pmod{121} \quad (21)$$

Step 3. We will now need to solve the congruences 20 and 21. We will employ Chinese Remainder Theorem for that. Suppose we have a system of congruences as below:

$$\begin{cases} x \equiv b_1 \pmod{n_1} \\ x \equiv b_2 \pmod{n_2} \\ x \equiv b_3 \pmod{n_3} \\ \vdots \\ x \equiv b_k \pmod{n_k} \end{cases} \quad (22)$$

CRT states that the above congruence has a unique modulo  $N = n_1 \cdot n_2 \cdot n_3 \dots n_k$  solution if each  $n_i$  are pairwise coprime and is given by:

$$\begin{aligned} x &= \sum_{i=1}^k b_i e_i (N/n_i) \\ \text{where } e_i &= (N/n_i)^{-1} \pmod{n_i} \end{aligned}$$

Restating our equations below:

$$\begin{cases} x \equiv 10 \pmod{27} \\ x \equiv 40 \pmod{121} \end{cases} \quad (23)$$

$$n_1 = 27, n_2 = 121$$

$$N = n = 27 \times 121 = 3267$$

$$b_1 = 10, b_2 = 40$$

$$e_1 = 121^{-1} \pmod{27} = 25$$

$$e_2 = 27^{-1} \pmod{121} = 9$$

$$\therefore x = 10 \times 25 \times 121 + 40 \times 9 \times 27 = 766 \pmod{3267}$$

$x = 766$  is the logarithm we wanted.

Below is the Maple Verification result:

```
find_log := proc(n, a, m)
description "Find log of a";
for i from 1 to n
do
if a · i mod n = m
then
return i;
fi;
enddo;
endproc;
```

```
1 >
2
```

```
ifactor(3267)
```

$$(3)^3 (11)^2 \quad (24)$$

```
n := 3^3 · 11^2;
```

$$n := 3267 \quad (25)$$

$g = x^{13}$ ,  $h = x^{157}$ ,  $n = 3267$ ;  $\langle g \rangle$  generates the group of order 3267;

Steps 1:

```
g1 := x13·112 mod 27;
```

$$g1 := x^7 \quad (26)$$

```
h1 := x157·112 mod 27;
```

$$h1 := x^{16} \quad (27)$$

We need to find the log of  $h1 = x^{16}$  in the cyclic group of order 27 generated by  $g1 = x^7$ . By trial and error, we get  $\log(h1) = 10$

```
find_log(27, 7, 16);
```

```
1 >
2
```

$$10 \quad (28)$$

So our first congruence is  $x1 = 10 \bmod 27$  — (1)

Step 2:

```
g2 := x13·33 mod 121;
```

$$g2 := x^{109} \quad (29)$$

```
h2 := x157·33 mod 121;
```

$$h2 := x^4 \quad (30)$$

We need to find the log of  $h2 = x^4$  in the cyclic group of order 121 generated by  $g2 = x^{109}$ ; using the proc find\_log above, it is = 118

*find\_log*(121, 109, 4);

$$40 \quad (31)$$

We get our second congruence as:  $x2 = 40 \pmod{121}$  — (2)

Hence we need to find the unique solution to  $x = 10 \pmod{27}$ , and  $x = 40 \pmod{121}$  using Chinese Remainder Theorem.

*with(NumberTheory);*

$$\begin{aligned} &[AreCoprime, CalkinWilfSequence, CarmichaelLambda, \\ &ChineseRemainder, ContinuedFraction, \\ &ContinuedFractionPolynomial, CyclotomicPolynomial, \\ &Divisors, FactorNormEuclidean, HomogeneousDiophantine, \\ &ImaginaryUnit, InhomogeneousDiophantine, IntegralBasis, \\ &InverseTotient, IsCyclotomicPolynomial, IsMersenne, \\ &IsSquareFree, IthFermat, IthMersenne, JacobiSymbol, \\ &JordanTotient, KroneckerSymbol, Landau, LargestNthPower, \\ &LegendreSymbol, Möbius, ModExtendedGCD, ModularLog, \\ &ModularRoot, ModularSquareRoot, Moebius, \\ &MultiplicativeOrder, Möbius, NearestLatticePoint, \\ &NextSafePrime, NumberOfIrreduciblePolynomials, \\ &NumberOfPrimeFactors,  $\Omega$ ,  $\Phi$ , PrimeCounting, PrimeFactors, \\ &PrimitiveRoot, PseudoPrimitiveRoot, QuadraticResidue, \\ &Radical, RepeatingDecimal, RootsOfUnity, \\ &SimplestRational, SumOfDivisors, SumOfSquares, ThueSolve, \\ &Totient,  $\lambda$ ,  $\mu$ ,  $\phi$ ,  $\pi$ ,  $\sigma$ ,  $\tau$ ,  $\varphi$ ] \end{aligned} \quad (32)$$

*ChineseRemainder*([10, 40], [27, 121]);

$$766 \quad (33)$$

6. Use the Pollard Rho method to verify your answer to the first example you were allocated in Question 4.

Name	b	n	a	Method
Ajeesh	47	71	21	BabyStepGiantStep
Ajeesh	24	53	26	BabyStepGiantStep
Ajeesh	$x^{41}$	343	$x^{11}$	Pohlig Hellmen
Ajeesh	$x^{157}$	3267	$x^{13}$	Pohlig Hellmen

Table 6: Table Listing the Problems Allocation to Individuals

**Answer:** The below table shows the execution of the Pollard Rho algorithm to generate the data table:

Pollard Rho Execution Data						
i	x1	a1	b1	x2	a2	b2
1	21	1	0	15	2	0
2	15	2	0	2	8	0
3	12	4	0	16	8	2
4	2	8	0	27	10	2
5	23	8	1	44	21	4
6	16	8	2	10	42	10
7	52	9	2	1	43	11
8	27	10	2	15	18	22
9	19	20	4	2	2	18
10	44	21	4	16	2	20
11	9	21	5	27	4	20
12	10	42	10	44	9	40
13	68	43	10	10	18	12
14	1	43	11	1	19	13

Table 7: Pollard Rho Execution Data

From the table, we can see that  $x1 = x2$  at the 14<sup>th</sup> iteration. The corresponding  $a1$ ,  $a2$ ,  $b1$ ,  $b2$  values are:  $i = 14$ ,  $a1 = 43$ ,  $a2 = 19$ ,  $b1 = 11$ ,  $b2 = 13$ .

Let  $t$  be the logarithm of  $b$ . Calculation of the logarithm is below:

$$\begin{aligned}
(b2 - b1).t &\equiv (a1 - a2) \pmod{p - 1} \\
&\implies (13 - 11).t \equiv (43 - 19) \pmod{70} \\
&\implies 2.t \equiv 24 \pmod{70}
\end{aligned} \tag{34}$$

Let  $d = \gcd(2, 70) = 2$

Divide Eqn. (34) by  $d$ , we get:  $t \equiv 12 \pmod{35}$

Since  $\gcd = 2$ , there are two solution to Eqn. (35) and the solutions are:

$$\left\{12, 12 + \frac{70}{2}\right\} = \{12, 47\} \tag{35}$$

When  $t = 47$  our congruence  $21^t \equiv 47 \pmod{71}$  is satisfied and hence the logarithm of 47 is 47. Thus we have verified solution of the problem in 4a.

## References

- [1] Dr Graham Taylor-Russell, *Lecture Notes - Number Theory for Cryptography*, London Metropolitan University
- [2] *MAS 335 Cryptography, Notes 7: Public-key cryptography*, Queen Mary, University Of London.