BRACU CP Workshop

Day 6

Ahnaf Shahriar Asif
Md Mubasshir Chowdhury

BRAC University, Dhaka

30 April 2025

Table of Contents

What will we learn ?

- Divisibility

What will we learn ?

- Divisibility
- Primes

What will we learn ?

- Divisibility
- Primes
- Modular Arithmetic

What will we learn ?

- Divisibility
- Primes
- Modular Arithmetic
- Primality testing

What will we learn ?

- Divisibility
- Primes
- Modular Arithmetic
- Primality testing
- Number theoretic functions

Divisibility

- Let a, b, and c be integers. Then:
  (i) If $b \mid a$ and $c \mid a$, then $bc \mid a$.
  (ii) If $b \mid a$ and $c \mid b$, then $c \mid a$.
  (iii) If $b \mid a$ and $c \mid b$, then $bc \mid ab$.
  (iv) If $c \mid a$ and $c \mid b$, then $c \mid (a+b)$ and $c \mid (a-b)$.

Finding divisors of n

- We can loop through all numbers and see if they divide n.

Finding divisors of n

- We can loop through all numbers and see if they divide n.
- Looping till $\sqrt{n}$ is enough.

Finding divisors of n

- We can loop through all numbers and see if they divide n.
- Looping till $\sqrt{n}$ is enough.
- We can find divisors of all numbers upto n in $\mathcal{O}(n \ln(n))$ time.

Finding divisors of n

- ```
  vector<vector<int>> divisors(n + 1);
  for (int i = 1; i <= n; i++) {
      for (int j = i; j <= n; j += i) {
          divisors[j].push_back(i);
      }
  }
  ```
  ```

Primes

- A prime p is greater than 1 and only divisible by 1 and p.

Primes

- A prime p is greater than 1 and only divisible by 1 and p.
- A composite number is a positive integer that is not prime.

Primes

- A prime p is greater than 1 and only divisible by 1 and p.
- A composite number is a positive integer that is not prime.
- Prime numbers help us calculate a lot of things easily.

Find number of divisors of n

- Given an integer $1 \leq n \leq 10^9$, find the number of divisors of n.

Solution: Number of divisors

- We can find the prime factorization of n.

Solution: Number of divisors

- We can find the prime factorization of n.
- If $n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$, then the number of divisors of n is given by:

$$d(n) = (e_1 + 1)(e_2 + 1) \cdots (e_k + 1)$$

Prime Factorization

- We can find the prime factorization of n in $\mathcal{O}(\sqrt{n})$ time.

Prime Factorization

- We can find the prime factorization of n in $\mathcal{O}(\sqrt{n})$ time.
- We can use a **sieve** to find all primes up to 10^6 in $\mathcal{O}(n \log(\log(n)))$ time.

Prime Factorization code

```
// vector<int> primes; (contains all primes upto sqrt(n)
vector<pair<int,int>> ppf(int n){
  vector<pair<int,int>> factors;
  for (int i = 0; primes[i] * primes[i] <= n; i++){
    int cnt = 0;
    while (n % primes[i] == 0){
      n /= primes[i];
      cnt++;
    }
    if (cnt > 0) factors.push_back({primes[i], cnt});
  }
  if (n > 1) factors.push_back({n, 1});
  return factors;
}
// if n = 12, it will return:
// {{2, 2}, {3, 1}} which means 12 = 2^2 * 3^1
```

Sieve

- We can use a sieve to find all primes up to 10^6 in $\mathcal{O}(n\log(\log(n)))$ time.

Sieve

- We can use a sieve to find all primes up to 10^6 in $\mathcal{O}(n \log(\log(n)))$ time.
- We can use the sieve to find the smallest prime factor of each number.

Sieve Simulation

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40

Sieve Code

```
bool is_composite[N];
vector<int> primes;

void sieve(int n){
  is_composite[1] = true;
  for(int i = 2; i <= n;i++){
    if(!is_composite[i]){
      primes.push_back(i);
      for(int j = i * i; j <= n;j+=i){
        is_composite[j] = true;
      }
    }
  }
}
```

HS08PAUL - A conjecture of Paul Erdos

Problem Link: https://www.spoj.com/problems/HS08PAUL/

- compute the number of **positive primes** not larger than a given integer n, which can be expressed in the form $x^2 + y^4$ for some integers x and y.

HS08PAUL - A conjecture of Paul Erdos

Problem Link: `https://www.spoj.com/problems/HS08PAUL/`

- compute the number of **positive primes** not larger than a given integer n, which can be expressed in the form $x^2 + y^4$ for some integers x and y.
- There will be 10^4 test cases, and $1 \leq n \leq 10^6$.

Rough - HS08PAUL

Solution

- First of all, we can find all the primes upto 10^7 using sieve. Let's say we've the **is_composite** array. We haven't pushed anything in the **primes** vector yet. Now, what we can do is go through all possible $x^2 + y^4$ and mark all the primes that are of that form. It is easy to notice that $x^2 \leq 10^7$ and $y^4 \leq 10^7$. Or in other words, $x \leq 10^4$ and $y \leq 60$. So, we can definitely loop through all possible x and y, and for each pair, we can check if $x^2 + y^4$ is prime. If it is, we can mark it in the **is_composite** array. Based on that array, we can generate a prefix sum array that can answer all the queries. And we can do all of these before the queries.

GCD, LCM

- GCD: Greatest Common Divisor

GCD, LCM

- GCD: Greatest Common Divisor
- LCM: Least Common Multiple

GCD, LCM

- GCD: Greatest Common Divisor
- LCM: Least Common Multiple
- GCD(a, b) $=$ GCD(b, a mod b)

GCD, LCM

- GCD: Greatest Common Divisor
- LCM: Least Common Multiple
- GCD(a, b) = GCD(b, a mod b)
- LCM(a, b) = (a * b) / GCD(a, b)

GCD

- GCD(a, b) can be computed in $\mathcal{O}(\log(\min(a, b)))$ time.

GCD

- GCD(a, b) can be computed in $\mathcal{O}(\log(\min(a, b)))$ time.
- Code:

```
int gcd(int a, int b){
  if (b == 0)return a;
  return gcd(b, a % b);
}
```

GCD

- GCD(a, b) can be computed in $\mathcal{O}(\log(\min(a, b)))$ time.
- Code:

```
int gcd(int a, int b){
  if (b == 0)return a;
  return gcd(b, a % b);
}
```

- if $a = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$ and $b = p_1^{f_1} \cdot p_2^{f_2} \cdots p_k^{f_k}$, then:

$$GCD(a, b) = p_1^{\min(e_1, f_1)} \cdot p_2^{\min(e_2, f_2)} \cdots p_k^{\min(e_k, f_k)}$$

LCM

- LCM(a, b) can be computed in $\mathcal{O}(\log(\min(a, b)))$ time.

LCM

- LCM(a, b) can be computed in $\mathcal{O}(\log(\min(a, b)))$ time.
- Code:

```
int lcm(int a, int b){
  return (a / gcd(a, b)) * b;
}
```

LCM

- LCM(a, b) can be computed in $\mathcal{O}(\log(\min(a, b)))$ time.
- Code:

```
int lcm(int a, int b){
  return (a / gcd(a, b)) * b;
}
```

- if $a = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$ and $b = p_1^{f_1} \cdot p_2^{f_2} \cdots p_k^{f_k}$, then:

$$LCM(a, b) = p_1^{\max(e_1, f_1)} \cdot p_2^{\max(e_2, f_2)} \cdots p_k^{\max(e_k, f_k)}$$

1968A - Maximize?

Problem link: https://codeforces.com/problemset/problem/1968/A

- You are given an integer x. Your task is to find any integer y such that $1 \leq y < x$ and $\gcd(x, y) + y$ is maximum possible. Note that if there is more than one y which satisfies the statement, you are allowed to find any.

1968A - Maximize?

Problem link: https://codeforces.com/problemset/problem/1968/A

- You are given an integer x. Your task is to find any integer y such that $1 \leq y < x$ and $\gcd(x, y) + y$ is maximum possible. Note that if there is more than one y which satisfies the statement, you are allowed to find any.
- $1 \leq t \leq 10^6$ and $1 \leq x \leq 10^{18}$

Rough

Rough - 1968A

Solution

Solution - 1968A

- Answer is $x - 1$.

Modular Arithmetic

1. $(a+b) \pmod m \equiv (a \mod m + b \mod m) \mod m$
2. $(a-b) \pmod m \equiv (a \mod m - b \mod m) \mod m$
3. $(a \cdot b) \pmod m \equiv (a \mod m \cdot b \mod m) \mod m$
4. $a^b \pmod m \equiv (a \mod m)^b \mod m$

Modular Exponension

- $a^b \pmod{m}$ can be computed in $\mathcal{O}(\log(b))$ time.

Modular Exponension

- $a^b \pmod{m}$ can be computed in $\mathcal{O}(\log(b))$ time.
- Code:

```
int mod_exp(int a, int b, int m){
  int res = 1;
  while (b > 0){
    if (b & 1)res = (res * a) % m;
    a = (a * a) % m;
    b >>= 1;
  }
  return res;
}
```

Mod Inverse

- $a^{-1} \pmod{m}$ can be computed in $\mathcal{O}(\log(m))$ time.

Mod Inverse

- $a^{-1} \pmod{m}$ can be computed in $\mathcal{O}(\log(m))$ time.
- Code:

```
int mod_inv(int a, int m){
  return mod_exp(a, m - 2, m);
}
```

NCR

- $C(n,r) = \frac{n!}{r!(n-r)!}$

NCR

- $C(n, r) = \frac{n!}{r!(n-r)!}$
- $C(n, r) \pmod{m}$ can be computed in $\mathcal{O}(n)$ time.

NCR

- $C(n, r) = \frac{n!}{r!(n-r)!}$
- $C(n, r) \pmod{m}$ can be computed in $\mathcal{O}(n)$ time.
- Code:

```
int ncr(int n, int r, int m){
  if (r > n)return 0;
  return (fact[n] * mod_inv(fact[r], m) % m *
}
```

Min = GCD

Codeforces 2084B

- Given an array a of length n.

Min = GCD

Codeforces 2084B

- Given an array a of length n.
- You can rearrange a in any order.

Min = GCD

Codeforces 2084B

- Given an array a of length n.
- You can rearrange a in any order.
- Determine if there is an arrangement of a such that there exists an integer $i (1 \le i \le n)$ such that

$$\min([a_1, a_2, \ldots, a_i]) = \gcd([a_{i+1}, a_{i+2}, \ldots, a_n])$$

Min = GCD

Codeforces 2084B

Sample Input

3

2 2 3

Sample Output

YES

Min = GCD

Codeforces 2084B

Sample Input

3

2 4 3

Sample Output

NO

Min = GCD

Codeforces 2084B

Sample Input

5
3 4 6 9 6

Sample Output

YES

Min = GCD

Codeforces 2084B

Observation:

- We need to divide the array into two parts, and determine which number goes where.

Min = GCD

Observation:

- We need to divide the array into two parts, and determine which number goes where.
- $\gcd(a) \leq \min(a)$.

Min = GCD

Codeforces 2084B

Observation:

- We need to divide the array into two parts, and determine which number goes where.
- $\gcd(a) \leq \min(a)$.
- The minimum number must be in the first part lets call it mn

Min = GCD

Codeforces 2084B

Observation:

- We need to divide the array into two parts, and determine which number goes where.
- $\gcd(a) \leq \min(a)$.
- The minimum number must be in the first part lets call it mn
- The second part must consists only multiples of mn.

Min = GCD

Codeforces 2084B

Observation:

- We need to divide the array into two parts, and determine which number goes where.
- $\gcd(a) \leq \min(a)$.
- The minimum number must be in the first part lets call it mn
- The second part must consists only multiples of mn.
- The second part should contain all multiples of mn.

Min = GCD

Codeforces 2084B

Observation:

- We need to divide the array into two parts, and determine which number goes where.
- $\gcd(a) \le \min(a)$.
- The minimum number must be in the first part lets call it mn
- The second part must consists only multiples of mn.
- The second part should contain all multiples of mn.
- Hence, the answer is YES if gcd all multiples of mn (except mn itself) equals to mn. Otherwise NO.

Simple Permutation

Codeforces 2089A

- Given an integer n. Construct a permutation $p_1, p_2, \ldots, p_n$ that follows the following rules:

Simple Permutation

Codeforces 2089A

- Given an integer n. Construct a permutation $p_1, p_2, \ldots, p_n$ that follows the following rules:
- Define $c_i = \lceil \frac{p_1+p_2+\ldots+p_i}{i} \rceil$.

Simple Permutation

Codeforces 2089A

- Given an integer n. Construct a permutation $p_1, p_2, \ldots, p_n$ that follows the following rules:
- Define $c_i = \lceil \frac{p_1 + p_2 + \ldots + p_i}{i} \rceil$.
- Then among $c_1, c_2, \ldots, c_n$, there are must be atleast $\lfloor \frac{n}{3} \rfloor - 1$ prime numbers.

Simple Permutation

Codeforces 2089A

Sample Input

5

Sample Output

2 1 3 4 5

Simple Permutation

Codeforces 2089A

Sample Input

6

Sample Output

3 2 1 4 5 6

Simple Permutation

Codeforces 2089A

Hint - Bertrand's Postulate:
For each positive integer $x \geq 1$, there exists at least one prime p such that $x < p < 2x$.

Simple Permutation

Codeforces 2089A

Solution:

- Choose a prime number p between $\lfloor \frac{n}{3} \rfloor$ and $\lceil \frac{2n}{3} \rceil$.

Simple Permutation

Codeforces 2089A

Solution:

- Choose a prime number p between $\lfloor \frac{n}{3} \rfloor$ and $\lceil \frac{2n}{3} \rceil$.
- Construct the permutation as follows:

$$p, p-1, p+1, p-2, p+2, \ldots$$

then fill the remaining numbers in any order.

Simple Permutation

Codeforces 2089A

Solution:

- Choose a prime number p between $\lfloor \frac{n}{3} \rfloor$ and $\lceil \frac{2n}{3} \rceil$.
- Construct the permutation as follows:

$$p, p-1, p+1, p-2, p+2, \ldots$$

  then fill the remaining numbers in any order.
- In this way, we have $c_1 = c_3 = c_5 = \ldots = c_{2\lfloor \frac{n}{3} \rfloor - 1} = p$

Problem about GCD

Codeforces 2043D

- Given three integers l, r, and G. ($1 \leq l \leq r \leq 10^{18}$, $1 \leq G \leq 10^{18}$)

Problem about GCD

Codeforces 2043D

- Given three integers l, r, and G. ($1 \leq l \leq r \leq 10^{18}$, $1 \leq G \leq 10^{18}$)
- Find two integers A and B such that:

$$l \leq A \leq B \leq r$$
$$\gcd(A, B) = G$$
$$|A - B| \text{ is maximized.}$$

If there are multiple such pairs, output the one with the smallest A.
If there is no such pair, output -1 -1.

Problem about GCD

Codeforces 2043D

Sample Input

4 12 2

Sample Output

4 10

Problem about GCD

Codeforces 2043D

Sample Input

4 8 3

Sample Output

-1 -1

Problem about GCD

Codeforces 2043D

Try solving if $G = 1$

- We will try $A = l$ and $B = r$. If $\gcd(A, B) = 1$, we found our answer.

Problem about GCD
Codeforces 2043D

Try solving if $G = 1$

- We will try $A = l$ and $B = r$. If $\gcd(A, B) = 1$, we found our answer.
- Then we will try $A = l$ and $B = r - 1$, and then $A = l + 1$ and $B = r$.

Problem about GCD

Codeforces 2043D

Try solving if $G = 1$

- We will try $A = l$ and $B = r$. If $\gcd(A, B) = 1$, we found our answer.
- Then we will try $A = l$ and $B = r - 1$, and then $A = l + 1$ and $B = r$.
- Then, we will try $(l, r - 2), (l + 1, r - 1), (l + 2, r)$ and so on.

Problem about GCD

Codeforces 2043D

Try solving if $G = 1$

- We will try $A = l$ and $B = r$. If $\gcd(A, B) = 1$, we found our answer.
- Then we will try $A = l$ and $B = r - 1$, and then $A = l + 1$ and $B = r$.
- Then, we will try $(l, r - 2), (l + 1, r - 1), (l + 2, r)$ and so on.
- We will keep trying until we find a pair (A, B) such that $\gcd(A, B) = 1$.

Problem about GCD

Codeforces 2043D

Try solving if $G = 1$

- We will try $A = l$ and $B = r$. If $\gcd(A, B) = 1$, we found our answer.
- Then we will try $A = l$ and $B = r - 1$, and then $A = l + 1$ and $B = r$.
- Then, we will try $(l, r - 2), (l + 1, r - 1), (l + 2, r)$ and so on.
- We will keep trying until we find a pair (A, B) such that $\gcd(A, B) = 1$.
- If $A > B$, we will stop (There is no solution).

Problem about GCD

Codeforces 2043D

But that seems slow?

Problem about GCD
Codeforces 2043D

Theorem
The algorithm will find atleast one pair of coprime pair between $[l, l+30)$ *and* $(r-30, r]$.

Problem about GCD

Codeforces 2043D

Theorem

The algorithm will find atleast one pair of coprime pair between $[l, l + 30)$ *and* $(r - 30, r]$.

Proof.

There are 900 pairs to be considered.

If a pair is not coprime, then there exists a prime number that divides both numbers.

Problem about GCD

Codeforces 2043D

Theorem

The algorithm will find atleast one pair of coprime pair between $[l, l + 30)$ *and* $(r - 30, r]$.

Proof.

There are 900 pairs to be considered.
If a pair is not coprime, then there exists a prime number that divides both numbers.
There will be atmost $15 \cdot 15 = 225$ pairs that are both divisible by 2. So we are left with 675 pairs

Problem about GCD

Codeforces 2043D

Theorem

The algorithm will find atleast one pair of coprime pair between $[l, l + 30)$ *and* $(r - 30, r]$.

Proof.

There are 900 pairs to be considered.
If a pair is not coprime, then there exists a prime number that divides both numbers.
There will be atmost $15 \cdot 15 = 225$ pairs that are both divisible by 2. So we are left with 675 pairs
There will be atmost $10 \cdot 10 = 100$ pairs that are both divisible by 3. So we are left with 575 pairs

Problem about GCD

Codeforces 2043D

Theorem

The algorithm will find atleast one pair of coprime pair between $[l, l+30)$ *and* $(r-30, r]$.

Proof.

There are 900 pairs to be considered.
If a pair is not coprime, then there exists a prime number that divides both numbers.
There will be atmost $15 \cdot 15 = 225$ pairs that are both divisible by 2. So we are left with 675 pairs
There will be atmost $10 \cdot 10 = 100$ pairs that are both divisible by 3. So we are left with 575 pairs
There wil be atmost $6 \cdot 6 = 36$ pairs that are both divisible by 5. So we are left with 539 pairs

Problem about GCD

Codeforces 2043D

Theorem

The algorithm will find atleast one pair of coprime pair between $[l, l+30)$ *and* $(r-30, r]$.

Proof.

There are 900 pairs to be considered.
If a pair is not coprime, then there exists a prime number that divides both numbers.
There will be atmost $15 \cdot 15 = 225$ pairs that are both divisible by 2. So we are left with 675 pairs
There will be atmost $10 \cdot 10 = 100$ pairs that are both divisible by 3. So we are left with 575 pairs
There wil be atmost $6 \cdot 6 = 36$ pairs that are both divisible by 5. So we are left with 539 pairs
If we keep doing this for upto prime 37, we will still have 450 pairs.
But now the prime numbers are greater than our range. $\square$

Problem about GCD

Codeforces 2043D

Theorem

The algorithm will find atleast one pair of coprime pair between $[l, l+30)$ *and* $(r-30, r]$.

Proof.

There are 30 numbers on the left side atmost. And each number has to be included in the pair. So one number has to be divisible by more than 15 prime numbers.

Problem about GCD

Codeforces 2043D

Theorem

The algorithm will find atleast one pair of coprime pair between $[l, l+30)$ *and* $(r-30, r]$.

Proof.

There are 30 numbers on the left side atmost. And each number has to be included in the pair. So one number has to be divisible by more than 15 prime numbers. But that would mean, that number is greater than 10^{18}. So there won't be any such number.

Problem about GCD

Codeforces 2043D

Theorem

The algorithm will find atleast one pair of coprime pair between $[l, l+30)$ *and* $(r-30, r]$.

Proof.

There are 30 numbers on the left side atmost. And each number has to be included in the pair. So one number has to be divisible by more than 15 prime numbers. But that would mean, that number is greater than 10^{18}. So there won't be any such number. Hence, there will be at least one pair of coprime numbers.

$\square$

Problem about GCD

Codeforces 2043D

If $G \neq 1$

- We can solve the problem for $\lceil \frac{l}{G} \rceil$ and $\lfloor \frac{r}{G} \rfloor$ then multiply the answer by G.

- Good Bye!