School of Electronics Engineering and Computer Science
Peking University

# Mathematics in Olympiad in Informatics

Frederica Haoyue Shi
hyshi@pku.edu.cn

January 20, 2017

# Content

# Introduction



English is **very helpful** to our future study and work. Please try to use English as much as possible, from now on.

Therefore, the slides are written in English.

If there's no specific instruction, the complexity is always time complexity.

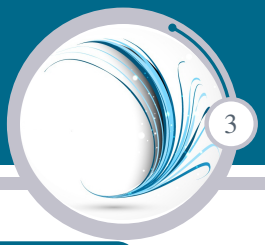
For the convenience of view, I recommend you to open it with Adobe Reader.

## Fundamental theorem of arithmetic

Every integer greater than 1 either is prime itself or is the product of prime numbers, and that this product is unique, up to the order of the factors. i.e.

$$\forall n > 1, n \in \mathbb{N}, n = p_1^{a_1} p_2^{a_2} \ldots p_n^{a_n}$$

where $p_k (1 \leq k \leq n), p_1 < p_2 < \ldots < p_n$ is prime.




## Fundamental theorem of arithmetic

Every integer greater than 1 either is prime itself or is the product of prime numbers, and that this product is unique, up to the order of the factors. i.e.

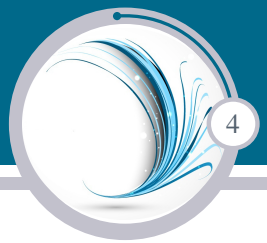
$$\forall n > 1, n \in \mathbb{N}, n = p_1^{a_1} p_2^{a_2} \ldots p_n^{a_n}$$

where $p_k (1 \leq k \leq n), p_1 < p_2 < \ldots < p_n$ is prime.

## Prime Number Theorem

$$\pi(N) \sim \frac{N}{\log(N)}$$

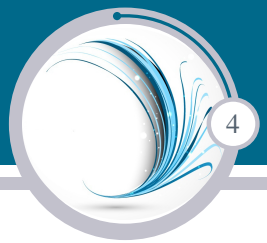
where $\pi(N)$ is the prime counting function, and $\log(N)$ is the natural logarithm of $N$. The equation means that for large enough $N$, the probability that a random natural number not greater than $N$ is prime is very close to $\frac{1}{\log(N)}$.




## How to check whether a number $N$ is prime?

Pseudo-code for prime number checking.

```
1: for i = 2 to √N do
2:     if i is divisor of N then
3:         return False
4:     end if
5: end for
```

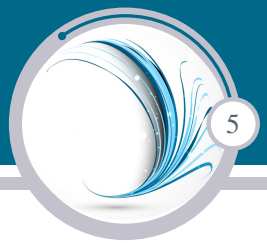



## How to check whether a number $N$ is prime?

Pseudo-code for prime number checking.

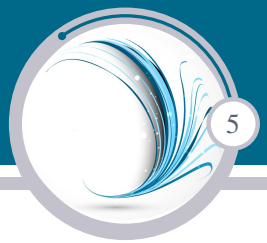
```
1: for i = 2 to √N do
2:     if i is divisor of N then
3:         return False
4:     end if
5: end for
```

The time complexity of such algorithm is $O(\sqrt{N})$.
The space complexity of such algorithm is $O(1)$.



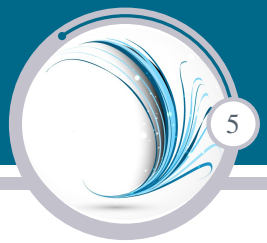

## How to generate a prime number list from 1 to $N$?

1. Check whether a number is prime one by one.




**How to generate a prime number list from 1 to $N$?**

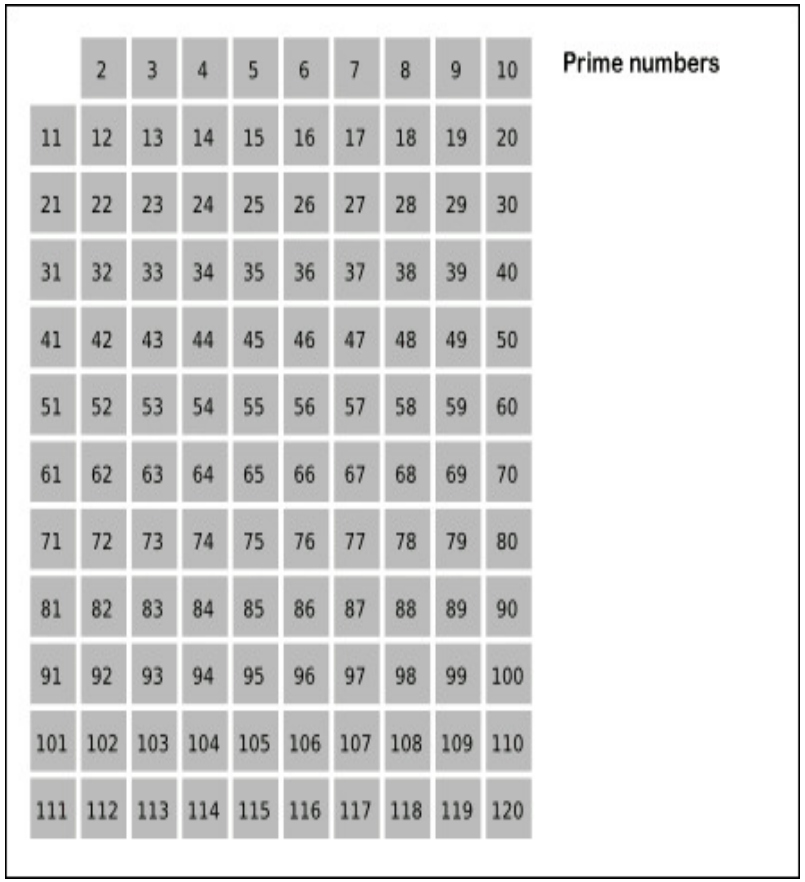
1. Check whether a number is prime one by one. $O(N\sqrt{N})$




**How to generate a prime number list from 1 to $N$?**

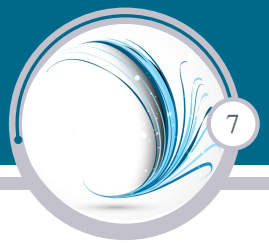
1. Check whether a number is prime one by one. $O(N\sqrt{N})$
2. Sieve of Eratosthenes

# Sieve of Eratosthenes




**How to generate a prime number list from 1 to $N$?**

1. Check whether a number is prime one by one. $O(N\sqrt{N})$
2. Sieve of Eratosthenes




**How to generate a prime number list from 1 to $N$?**

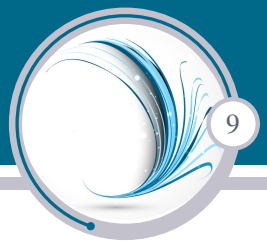
1. Check whether a number is prime one by one. $O(N\sqrt{N})$
2. Sieve of Eratosthenes $O(N\log(N))$


**How to generate a prime number list from 1 to $N$?**

1. Check whether a number is prime one by one. $O(N\sqrt{N})$
2. Sieve of Eratosthenes $O(N\log(N))$
3. Linear Sieve

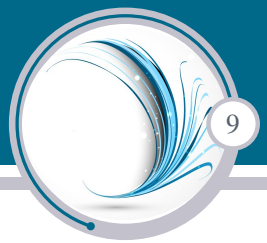

Pseudo-code for linear sieve.

```
1:  for i = 2 to N do
2:      set IsPrime[i] = True
3:  end for
4:  for i = 2 to N do
5:      if IsPrime[i] then
6:          push i into PrimeNumberList
7:      end if
8:      j = 0
9:      while j < CurrentPrimeNumber and i·PrimeNumberList[j]< N do
10:         IsPrime[i·PrimeNumberList[j]] = False
11:         if PrimeNumberList[j] is divisor of i then
12:             break
13:         end if
14:         increase j
15:     end while
16: end for
```




## Linear Sieve

The time complexity of such algorithm is $O(N)$.
The space complexity of such algorithm is $O(N)$.

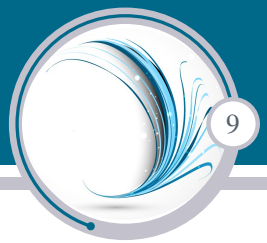



## Linear Sieve

The time complexity of such algorithm is $O(N)$.
The space complexity of such algorithm is $O(N)$.

**Practice: POJ 2689**

Given $L, U (1 \leq L < U < 2^{31}, U - L \leq 1,000,000)$, you are to find the two adjacent primes $C_1$ and $C_2 (L \leq C_1 < C_2 \leq U)$ that are closest (i.e. $C_2 - C_1$ is the minimum), and the two adjacent primes $D_1$ and $D_2 (L \leq D_1 < D_2 \leq U)$ that are the most distant (i.e. $D_2 - D_1$ is the maximum).




## Linear Sieve

The time complexity of such algorithm is $O(N)$.
The space complexity of such algorithm is $O(N)$.

**Practice: POJ 2689**

Given $L, U (1 \leq L < U < 2^{31}, U - L \leq 1,000,000)$, you are to find the two adjacent primes $C_1$ and $C_2 (L \leq C_1 < C_2 \leq U)$ that are closest (i.e. $C_2 - C_1$ is the minimum), and the two adjacent primes $D_1$ and $D_2 (L \leq D_1 < D_2 \leq U)$ that are the most distant (i.e. $D_2 - D_1$ is the maximum).

Hint: You are definitely required to generate a prime number list in the range of $[L, U]$. But how?

# Optional: Mathematical Induction



Mathematical induction is a mathematical proof technique used to prove a given statement about any well-ordered set. Most commonly, it is used to establish statements for the set of all natural numbers.

# Optional: Mathematical Induction



Mathematical induction is a mathematical proof technique used to prove a given statement about any well-ordered set. Most commonly, it is used to establish statements for the set of all natural numbers.

It contains two steps:

1. **base case**, to prove the given statement for the least element in the well-ordered set.

# Optional: Mathematical Induction



Mathematical induction is a mathematical proof technique used to prove a given statement about any well-ordered set. Most commonly, it is used to establish statements for the set of all natural numbers.

It contains two steps:

1. **base case**, to prove the given statement for the least element in the well-ordered set.
2. **inductive step**, to prove that, if the statement is assumed to be true for any element, then it must be true for the next element as well.

# Optional: Mathematical Induction


## An Example

Prove $\sum_{i=0}^{n} = \frac{n(n+1)}{2}, \forall n \in \mathbb{N}$

# Optional: Mathematical Induction


## An Example

Prove $\sum_{i=0}^{n} = \frac{n(n+1)}{2}, \forall n \in \mathbb{N}$

*Proof* For the case $n = 0$, we have $\sum_{i=0}^{n} = \frac{n(n+1)}{2} = 0$.


## An Example

Prove $\sum_{i=0}^{n} = \frac{n(n+1)}{2}, \forall n \in \mathbb{N}$

*Proof* For the case $n = 0$, we have $\sum_{i=0}^{n} = \frac{n(n+1)}{2} = 0$.

Assume that $\sum_{i=0}^{n} = \frac{n(n+1)}{2}$ when $n = k$,

then $\sum_{i=0}^{k+1} = \frac{k(k+1)}{2} + k + 1 = \frac{k(k+1)+2(k+1)}{2} = \frac{(k+1)(k+2)}{2}$.

Hence, the statement is true when $n = k + 1$.

# Optional: Mathematical Induction



## An Example

Prove $\sum_{i=0}^{n} = \frac{n(n+1)}{2}, \forall n \in \mathbb{N}$

*Proof* For the case $n = 0$, we have $\sum_{i=0}^{n} = \frac{n(n+1)}{2} = 0$.

Assume that $\sum_{i=0}^{n} = \frac{n(n+1)}{2}$ when $n = k$,

then $\sum_{i=0}^{k+1} = \frac{k(k+1)}{2} + k + 1 = \frac{k(k+1)+2(k+1)}{2} = \frac{(k+1)(k+2)}{2}$.

Hence, the statement is true when $n = k + 1$.

Therefore, $\sum_{i=0}^{n} = \frac{n(n+1)}{2}, \forall n \in \mathbb{N}$. #


## Euclidean Algorithm for Greatest Common Divisor(GCD)

Pseudo-code for Euclidean Algorithm.

```
1: function GCD(integer a,b)
2: if b == 0 then
3:     return a
4: else
5:     return GCD(b, a mod b)
6: end if
```


## Euclidean Algorithm for Greatest Common Divisor(GCD)

Pseudo-code for Euclidean Algorithm.

```
1: function GCD(integer a,b)
2: if b == 0 then
3:     return a
4: else
5:     return GCD(b, a mod b)
6: end if
```

Most of you know how, but why?


## Euclidean Algorithm for Greatest Common Divisor(GCD)

*Proof*

For any two integers $a, b$, $a$ can be written as

$$a = bq + r$$

where $q, r$ are non-negative integers, $0 \leq r < b$.


## Euclidean Algorithm for Greatest Common Divisor(GCD)

*Proof*

For any two integers $a, b$, $a$ can be written as

$$a = bq + r$$

where $q, r$ are non-negative integers, $0 \leq r < b$.

For all intergers $d$, if $d|a$ and $d|b$, then

$$d|bq \Rightarrow d|(a - bq) \Rightarrow d|r$$

## Euclidean Algorithm for Greatest Common Divisor(GCD)

*Proof*

For any two integers $a, b$, $a$ can be written as

$$a = bq + r$$

where $q, r$ are non-negative integers, $0 \leq r < b$.
For all intergers $d$, if $d|a$ and $d|b$, then

$$d|bq \Rightarrow d|(a - bq) \Rightarrow d|r$$

Namely, if $d$ is a common divisor of $a$ and $b$, then $d$ is a common divisor of $b$ and $r$, and vice versa.

## Euclidean Algorithm for Greatest Common Divisor(GCD)

*Proof*

For any two integers $a, b$, $a$ can be written as

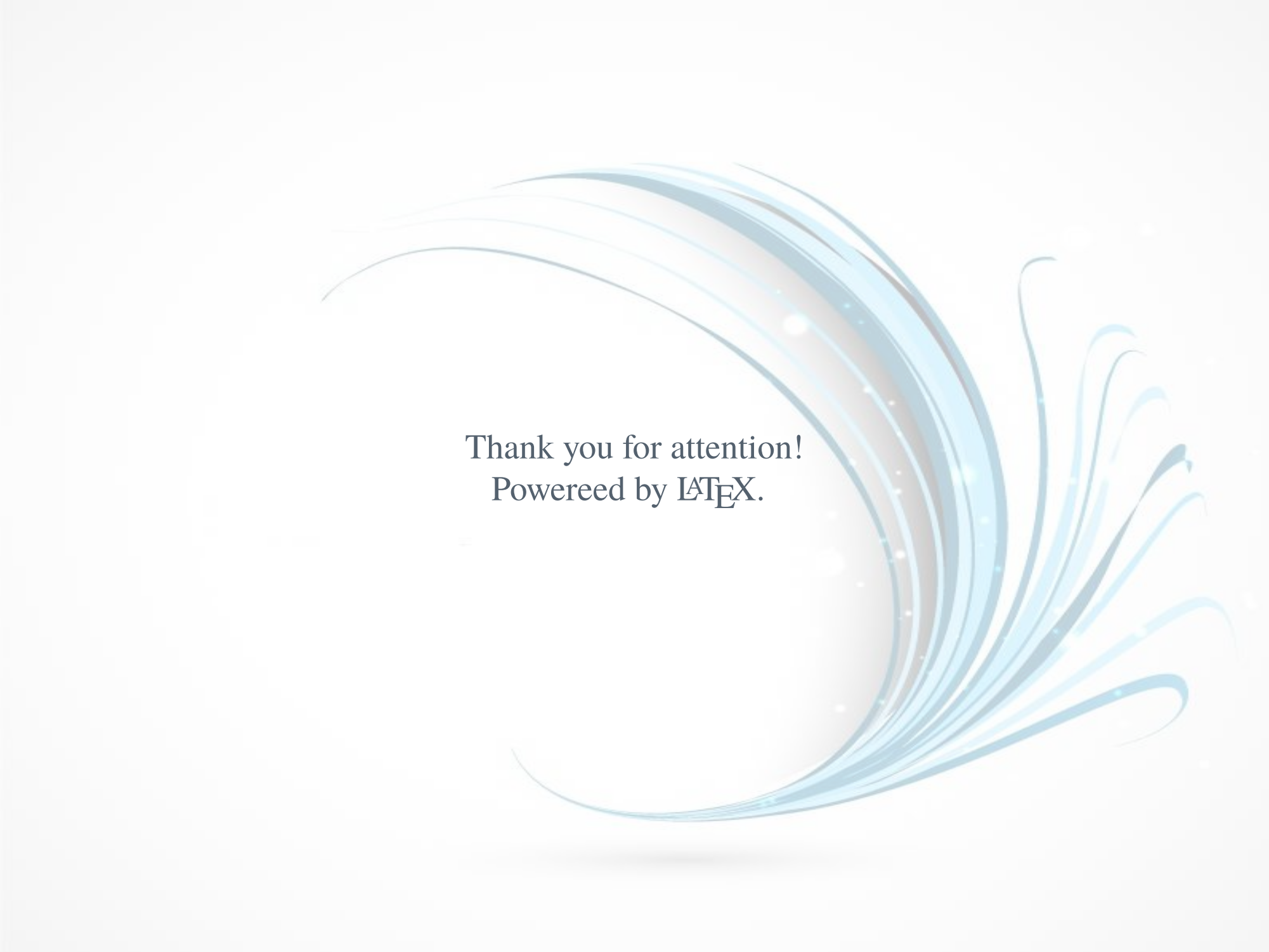
$$a = bq + r$$

where $q, r$ are non-negative integers, $0 \leq r < b$.
For all intergers $d$, if $d|a$ and $d|b$, then

$$d|bq \Rightarrow d|(a - bq) \Rightarrow d|r$$

Namely, if $d$ is a common divisor of $a$ and $b$, then $d$ is a common divisor of $b$ and $r$, and vice versa.
Hence, GCD of $a$ and $b$ (denoted as $(a, b)$) equals $(b, r)$.

## Euler's totient function


Thank you for attention!
Powereed by LaTeX.