Sieve of Eratosthenes

Find All Prime Numbers - The Fast Way

Mausam Kar

February 4, 2026

Contents

1 What is Sieve of Eratosthenes?

The Big Idea

The FASTEST way to find ALL prime numbers up to N.

Instead of checking each number → We ELIMINATE multiples!
What survives = Prime numbers

1.1 What is a Prime Number?

Prime Number = A number that can only be divided by 1 and itself

Examples:

- 2 is prime (only divisible by 1 and 2)
- 3 is prime (only divisible by 1 and 3)
- 5 is prime (only divisible by 1 and 5)
- 4 is NOT prime (divisible by 1, 2, and 4)
- 6 is NOT prime (divisible by 1, 2, 3, and 6)

1.2 The Simple Strategy

One-Line Strategy

Start with the first prime and keep deleting its multiples.
What survives are primes!

Think of it like this:

1. Write all numbers from 2 to N
2. Circle 2 (it's prime), cross out all multiples of 2
3. Circle next uncrossed number (3), cross out its multiples
4. Circle next uncrossed number (5), cross out its multiples
5. Keep going until done

6. All circled numbers = primes!

2 Step-by-Step Visual Example

Let's find all primes up to 30.

2.1 Step 0: Start with All Numbers

Write numbers from 2 to 30:

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	

Assume all are prime (for now).

2.2 Step 1: Start with p = 2

2 is prime! Circle it.
Now cross out all multiples of 2 (except 2 itself):
Multiples of 2: 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30

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	

2.3 Step 2: Next uncrossed number = 3

3 is prime! Circle it.
Cross out multiples of 3: 6, 9, 12, 15, 18, 21, 24, 27, 30
(Note: 6, 12, 18, 24, 30 already crossed by 2)

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	

2.4 Step 3: Next uncrossed = 5

5 is prime! Circle it.
Cross out multiples of 5: 10, 15, 20, 25, 30
(Most already crossed)

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	

2.5 Step 4: STOP!

We can stop now because:

$\sqrt{30} \approx 5.47$

Next number would be 7, but $7 \times 7 = 49 > 30$

All remaining uncrossed numbers are prime!

2.6 Final Result

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	

Primes up to 30:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29

Total: 10 primes

3 Why Start at p $\times$ p? (IMPORTANT!)

3.1 The Big Question

When we process prime p, why do we start crossing out from $p \times p$?

Why not start from $2 \times p$ or p itself?

3.2 The Simple Answer

Key Insight

All smaller multiples of p have ALREADY been crossed out by smaller primes!
So $p \times p$ is the FIRST multiple we need to handle.

3.3 Example with p = 5

When we process p = 5, let's look at its multiples:

Multiple	Why already crossed?	Status
$5 \times 2 = 10$	Crossed by p=2 (10 is multiple of 2)	Already done
$5 \times 3 = 15$	Crossed by p=3 (15 is multiple of 3)	Already done
$5 \times 4 = 20$	Crossed by p=2 (20 is multiple of 2)	Already done
$5 \times 5 = 25$	NOT crossed yet!	FIRST NEW ONE

So we start from $5 \times 5 = 25$ to avoid duplicate work!

3.4 Example with p = 7

Multiple	Why already crossed?	Status
$7 \times 2 = 14$	Crossed by p=2	Already done
$7 \times 3 = 21$	Crossed by p=3	Already done
$7 \times 4 = 28$	Crossed by p=2	Already done
$7 \times 5 = 35$	Crossed by p=5	Already done
$7 \times 6 = 42$	Crossed by p=2	Already done
$7 \times 7 = 49$	NOT crossed yet!	FIRST NEW ONE

3.5 Visual Proof

Why $p \times p$ is the Starting Point

p=5

5×2=10 ← Already crossed by 2
5×3=15 ← Already crossed by 3
5×4=20 ← Already crossed by 2

5×5=25 ← **FIRST uncrossed! Start here!**

3.6 The Formula

For prime p:
Start crossing from: $p \times p$
Then continue: $p \times p + p$, $p \times p + 2p$, $p \times p + 3p$, ...
Or in code:

```
for (multiple = p*p; multiple <= N; multiple += p)
    mark[multiple] = not prime
```

3.7 p × p vs p × 2: Why p × p is BETTER

Simple Comparison

Starting at p × p = OPTIMAL (smart!)
Starting at p × 2 = WORST CASE (wasteful!)

Why?
If you start at $p \times 2$, you do EXTRA WORK that was already done!

3.7.1 Example: p = 5

Starting at p × 2 = 10:

Multiple	Already handled by?	Wasted?
$5 \times 2 = 10$	Prime 2 ($10 = 2 \times 5$)	WASTE
$5 \times 3 = 15$	Prime 3 ($15 = 3 \times 5$)	WASTE
$5 \times 4 = 20$	Prime 2 ($20 = 2 \times 10$)	WASTE
$5 \times 5 = 25$	NOT handled yet	USEFUL
$5 \times 6 = 30$	Prime 2 ($30 = 2 \times 15$)	WASTE

Work done:

- Total multiples checked: 5
- Actually needed: 1 (only 25!)
- Wasted work: 4 operations

Starting at $\mathbf{p \times p = 25}$:

- Start directly at 25
- NO wasted work!
- Only mark numbers not already crossed

3.8 The Big Picture

Summary:

$\mathbf{p \times p}$ (optimal):

- Skips work done by smaller primes
- Starts exactly where needed
- Makes sieve FAST!

$\mathbf{p \times 2}$ (worst case):

- Repeats work already done
- Checks already-crossed numbers
- Makes sieve SLOW!

Bottom line: $p \times p$ is the optimization that makes Sieve super fast!

4 All Iterations Shown (N = 50)

Let's see EVERY step for finding primes up to 50.

4.1 Which p values do we check?

We check p = 2, 3, 4, 5, 6, 7...

But we only ACT when p is still marked prime.

We STOP when $p \times p > 50$ (i.e., p ¿ 7)

4.2 Complete Iteration Table

p	**Is p prime?**	**p × p**	**Action**
2	YES	4	Cross: 4, 6, 8, 10, 12, ..., 50
3	YES	9	Cross: 9, 12, 15, 18, 21, ..., 48
4	NO	-	Skip (already crossed by 2)
5	YES	25	Cross: 25, 30, 35, 40, 45, 50
6	NO	-	Skip (crossed by 2 and 3)
7	YES	49	Cross: 49 (only one!)
8+	-	> 50	STOP (8×8=64 ¿ 50)

4.3 Detailed View - p = 2

Start: $p \times p = 2 \times 2 = 4$

Cross out: 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50

Pattern: Every 2nd number starting from 4

4.4 Detailed View - p = 3

Start: $p \times p = 3 \times 3 = 9$

Cross out: 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48

Pattern: Every 3rd number starting from 9

Note: 6 already crossed by 2, so we skip it!

4.5 Detailed View - p = 5

Start: $p \times p = 5 \times 5 = 25$

Cross out: 25, 30, 35, 40, 45, 50

Pattern: Every 5th number starting from 25

Note: 10, 15, 20 already crossed!

4.6 Detailed View - p = 7

Start: $p \times p = 7 \times 7 = 49$

Cross out: 49

Next would be: 49 + 7 = 56 ¿ 50 (stop)

4.7 Final Primes up to 50

All primes:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47
Total: 15 primes

5 Simple Pseudocode

5.1 Algorithm in Plain Words

1. Create array of size N+1, mark all as prime

2. Mark 0 and 1 as not prime

3. Loop p from 2 to $\sqrt{N}$:

   - If p is still marked prime:
     - Cross out all multiples starting from $p \times p$

4. All numbers still marked prime → Print them!

5.2 Pseudocode

```
SIEVE(N):
    1. Create array prime[0...N]
    2. Set all prime[i] = TRUE

    3. prime[0] = FALSE
    4. prime[1] = FALSE

    5. For p = 2 to N:
    6.     If prime[p] == TRUE:
    7.         For multiple = p*p to N step p:
    8.             prime[multiple] = FALSE

    9. For i = 2 to N:
   10.     If prime[i] == TRUE:
   11.         Print i
```

5.3 Line-by-Line Explanation

Lines 1-2: Create array and assume all are prime initially
Lines 3-4: 0 and 1 are not prime (by definition)
Line 5: Only loop up to $\sqrt{N}$ (optimization!)
Line 6: Check if p is still prime (not crossed out)
Line 7: Start from $p \times p$ (first uncrossed multiple)
Line 8: Mark as not prime
Lines 9-11: Print all remaining primes

6 Java Implementation

6.1 Simple Version

Listing 1: Sieve of Eratosthenes - Basic Version

```java
import java.util.*;

public class SieveSimple {

    public static void sieve(int n) {
        // Step 1: Create boolean array (true = prime)
        boolean[] prime = new boolean[n + 1];

        // Step 2: Assume all numbers are prime
        Arrays.fill(prime, true);

        // Step 3: 0 and 1 are not prime
        prime[0] = false;
        prime[1] = false;

        // Step 4: Loop for p from 2 to sqrt(n)
        for (int p = 2; p * p <= n; p++) {

            // Step 5: If p is still prime
            if (prime[p] == true) {

                // Step 6: Cross out multiples starting from p*p
                for (int multiple = p * p; multiple <= n; multiple += p) {
                    prime[multiple] = false;  // Mark as not prime
                }
            }
        }

        // Step 7: Print all primes
        System.out.println("Primes up to " + n + ":");
        for (int i = 2; i <= n; i++) {
            if (prime[i]) {
                System.out.print(i + " ");
            }
        }
        System.out.println();
    }

    public static void main(String[] args) {
        sieve(30);
        // Output: 2 3 5 7 11 13 17 19 23 29
    }
}
```

7 Time & Space Complexity

7.1 Time Complexity

Time Complexity

O(N log log N)
This is VERY fast!

Why so fast?

- We only loop up to $\sqrt{N}$
- For each p, we don't check—we just mark multiples
- Starting from $p \times p$ saves a lot of work

Comparison:

Method	Time	For N=1,000,000
Check each number individually	O(NN)	Very slow
Sieve of Eratosthenes	O(N log log N)	Super fast!

7.2 Space Complexity

Space Complexity

O(N)
We need an array of size N+1

8 Quick Summary

8.1 The Algorithm

Sieve of Eratosthenes - One-Line Idea:
Cross out multiples of each prime → What survives are primes!

Steps:

1. Mark all numbers as prime
2. For each prime p (starting from 2):
   - Cross out all multiples starting from $p \times p$
3. Numbers still marked = primes!

8.2 Why p × p?

Key Optimization:
All multiples smaller than $p \times p$ have already been crossed out by smaller primes.
So $p \times p$ is the FIRST new multiple we need to handle!
Example:

- For p=5: Start at 25 (not 10, 15, 20)
- For p=7: Start at 49 (not 14, 21, 28, ...)

8.3 Code Template

```
boolean[] prime = new boolean[n + 1];
Arrays.fill(prime, true);
prime[0] = prime[1] = false;

for (int p = 2; p * p <= n; p++) {
    if (prime[p]) {
        for (int m = p*p; m <= n; m += p) {
            prime[m] = false;
        }
    }
}
```

8.4 Complexity

Time: O(N log log N) — Very fast!
Space: O(N) — Need array of size N