Segmented Sieve

Detailed & Simple

Mausam Kar

February 4, 2026

Contents

1 The Problem

Why Normal Sieve Fails

Normal Sieve needs array of size N.
For N = 1,000,000,000 (1 billion) $\rightarrow$ Need 1 billion array!
Memory can't handle this!

Example:

- Find primes up to 100 $\rightarrow$ Array[100] Easy!
- Find primes up to 10^{12} $\rightarrow$ Array[10^{12}] Impossible!

Solution: Don't create one huge array. Process in SMALL CHUNKS!

2 Algorithm - Detailed Explanation

2.1 The Core Idea

Two-Step Process

Step 1: Find small "base" primes up to N
Step 2: Use those base primes to check larger numbers in segments

2.2 Why Only N?

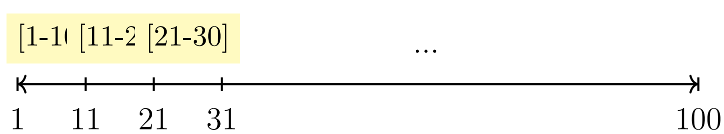
Key Fact: Any composite number N must have at least one prime factor N

Example:
Number 35 is composite: $35 = 5 \times 7$
35 5.9
One factor (5) is 35
This means: To check if numbers up to N are prime, we only need primes up to N!

2.3 How Segments Work - Visual

Task: Find primes from 1 to 100

Step 1: Get base primes up to $100 = 10$
Base primes: **2, 3, 5, 7**
Step 2: Divide 1-100 into segments:



Step 3: For each segment:

- Create small array (size 10, not 100!)
- Use base primes [2,3,5,7] to mark composites
- Find primes in this segment
- Move to next segment

Memory saved: Only need array of size 10, not 100!

3 Detailed Walkthrough

Example: Find primes in [20, 40]

3.1 Step 1: Get Base Primes

40 6.3

Use normal sieve to find primes up to 6: **2, 3, 5**

3.2 Step 2: Create Segment Array

Range [20, 40] has 40 - 20 + 1 = 21 numbers

Create array of size 21:

Index	0	1	2	3	...	20
Number	20	21	22	23	...	40
Initially	T	T	T	T	...	T

Important: Index Number!

Index 0 represents number 20

Index 1 represents number 21, etc.

3.3 Step 3: Use Base Prime p=2

Question: Which multiples of 2 are in [20, 40]?

First multiple: 20 (because $20 = 2 \times 10$)

Mark all multiples: 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40

20	21	22	23	24	25	26
F	T	F	T	F	T	F

(F = false = composite, T = true = maybe prime)

3.4 Step 4: Use Base Prime p=3

How to find first multiple of 3 in [20, 40]?

Method: $20 \div 3 = 6.666...$

Round UP to 7

First multiple: $7 \times 3 = 21$

Mark: 21, 24, 27, 30, 33, 36, 39

3.5 Step 5: Use Base Prime p=5

First multiple: 20 (because $20 = 5 \times 4$)

Mark: 20, 25, 30, 35, 40

3.6 Step 6: Result

All numbers still TRUE are prime:

Primes in [20, 40]: 23, 29, 31, 37
Total: 4 primes

4 Pseudocode with Explanation

```
SEGMENTED_SIEVE(L, R):

1. limit = sqrt(R)
2. basePrimes = normalSieve(limit)

3. segmentSize = limit
4. For low = L to R step segmentSize:
5.     high = min(low + segmentSize - 1, R)

6.     Create mark[0...high-low]
7.     Fill all with TRUE

8.     For each prime p in basePrimes:
9.         start = findFirstMultiple(p, low)

10.        For j = start to high step p:
11.            mark[j - low] = FALSE

12.    Print all i where mark[i] == TRUE

---

findFirstMultiple(p, low):
    if p*p >= low:
        return p*p
    else:
        return ceil(low/p) * p
```

4.1 Line-by-Line Explanation

Line 1: Calculate R to find how many base primes we need

Line 2: Get all primes up to R using normal sieve

Line 4-5: Loop through segments. Each segment starts at `low` and ends at `high`

Line 6-7: Create small array for just this segment

Line 9: Find first multiple of p in this segment (detailed below)

Line 11: `mark[j - low]` converts number j to array index

Example: Number 23 in segment [20-30] → Index = 23 - 20 = 3

4.2 The Tricky Part: Finding First Multiple

Why is this important?

We want to start marking from the FIRST multiple of p in [low, high]

Two cases:

Case 1: If $p \times p \geq$ low $\rightarrow$ Use $p \times p$ (same as normal sieve)

Case 2: If $p \times p <$ low $\rightarrow$ Calculate first multiple

Formula: $\lceil low/p \rceil \times p$

Example:

Segment [20, 40], p = 3

$3 \times 3 = 9 < 20 \rightarrow$ Can't use $p \times p$

Calculate: $\lceil 20/3 \rceil \times 3 = 7 \times 3 = 21$

5 Java Code - Simple & Working

Listing 1: Segmented Sieve - Complete Working Code

```java
import java.util.*;

public class SegmentedSieve {

    // Step 1: Get base primes using normal sieve
    public static List<Integer> getBasePrimes(int limit) {
        boolean[] prime = new boolean[limit + 1];
        Arrays.fill(prime, true);
        prime[0] = prime[1] = false;

        // Normal sieve
        for (int p = 2; p * p <= limit; p++) {
            if (prime[p]) {
                for (int i = p * p; i <= limit; i += p) {
                    prime[i] = false;
                }
            }
        }

        // Collect primes
        List<Integer> primes = new ArrayList<>();
        for (int i = 2; i <= limit; i++) {
            if (prime[i]) primes.add(i);
        }
        return primes;
    }

    // Step 2: Find primes in range [L, R] using segments
    public static void findPrimes(long L, long R) {
        // Get base primes
        int limit = (int) Math.sqrt(R);
        List<Integer> basePrimes = getBasePrimes(limit);

        System.out.println("Base primes: " + basePrimes);
        System.out.println("\nPrimes in [" + L + ", " + R + "]:");

        // Process in segments
        long segSize = limit;

        for (long low = L; low <= R; low += segSize) {
            long high = Math.min(low + segSize - 1, R);

            // Create array for this segment only
```

```java
            int size = (int)(high - low + 1);
            boolean[] isPrime = new boolean[size];
            Arrays.fill(isPrime, true);

            // Use each base prime to mark composites
            for (int p : basePrimes) {
                // Find first multiple of p in [low, high]
                long start;

                if ((long)p * p >= low) {
                    // Case 1: Use p*p
                    start = (long)p * p;
                } else {
                    // Case 2: Calculate first multiple
                    start = (low / p) * p;
                    if (start < low) start += p;
                }

                // Mark all multiples in this segment
                for (long j = start; j <= high; j += p) {
                    // Convert number j to array index
                    isPrime[(int)(j - low)] = false;
                }
            }

            // Print primes in this segment
            for (int i = 0; i < size; i++) {
                if (isPrime[i] && (low + i) > 1) {
                    System.out.print((low + i) + " ");
                }
            }
        }
        System.out.println();
    }

    public static void main(String[] args) {
        // Example 1
        findPrimes(20, 50);

        System.out.println("\n" + "=".repeat(50) + "\n");

        // Example 2: Large range
        findPrimes(1000, 1100);
    }
}
```

6 Complete Trace Example

Find primes in [20, 30]

6.1 Setup

30 5.5 → Base primes: [2, 3, 5]
Segment size: 30 - 20 + 1 = 11

6.2 Initial State

Index	0	1	2	3	4	5	6	7	8	9	10
Number	20	21	22	23	24	25	26	27	28	29	30
State	T	T	T	T	T	T	T	T	T	T	T

6.3 After p=2

First multiple: 20
Mark: 20, 22, 24, 26, 28, 30

20	21	22	23	24	25	26	27	28	29	30
F	T	F	T	F	T	F	T	F	T	F

6.4 After p=3

First multiple: 21
Mark: 21, 24, 27, 30

20	21	22	23	24	25	26	27	28	29	30
F	F	F	T	F	T	F	F	F	T	F

6.5 After p=5

First multiple: 20
Mark: 20, 25, 30

20	21	22	23	24	25	26	27	28	29	30
F	F	F	T	F	F	F	F	F	T	F

6.6 Final Result

Numbers still TRUE: **23, 29**

Primes in [20, 30]: 23, 29
Total: 2 primes

7 Summary

Segmented Sieve - Process in chunks

Steps:

1. Get base primes up to N
2. Divide range into small segments
3. For each segment:
    - Create small array
    - Use base primes to mark composites
    - Find primes

Key formulas:

- Base primes: up to R
- Segment size: R
- Index: number - low
- First multiple: $\max(p \times p, \lceil low/p \rceil \times p)$

Use when:

- N is very large (¿ 10)
- Memory limited
- Range queries [L, R]