**南 京 师 范 大 学**

**《密码学基础》实验报告**



**题 目： 素性检验**

**姓 名： 刘浩天**

**学 号： 19220211**

**实验环境： eclipse**

**时 间： 2025年4月8日**

1. **实验目的**

1.熟悉素性检验方法：

通过实验了解并掌握常用的素性检验算法，特别是埃拉托斯特尼筛法（Sieve of Eratosthenes）的原理和应用。

2.编程实践：

运用Java语言实现埃拉托斯特尼筛法，加深对算法逻辑的理解，提升编程能力。

3.算法分析：

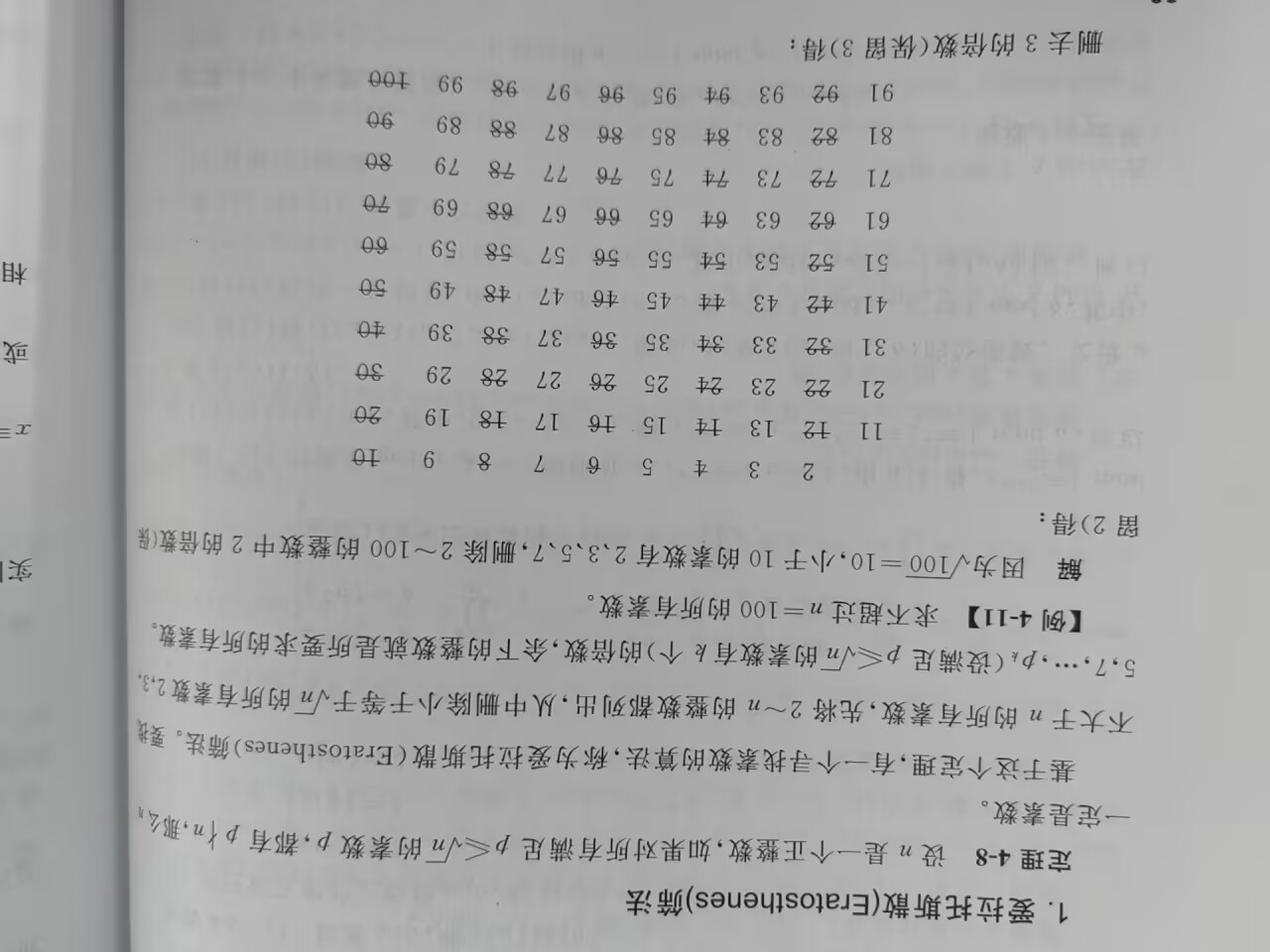
通过实验过程分析算法的时间复杂度和空间复杂度，理解其在不同规模数据下的性能表现。

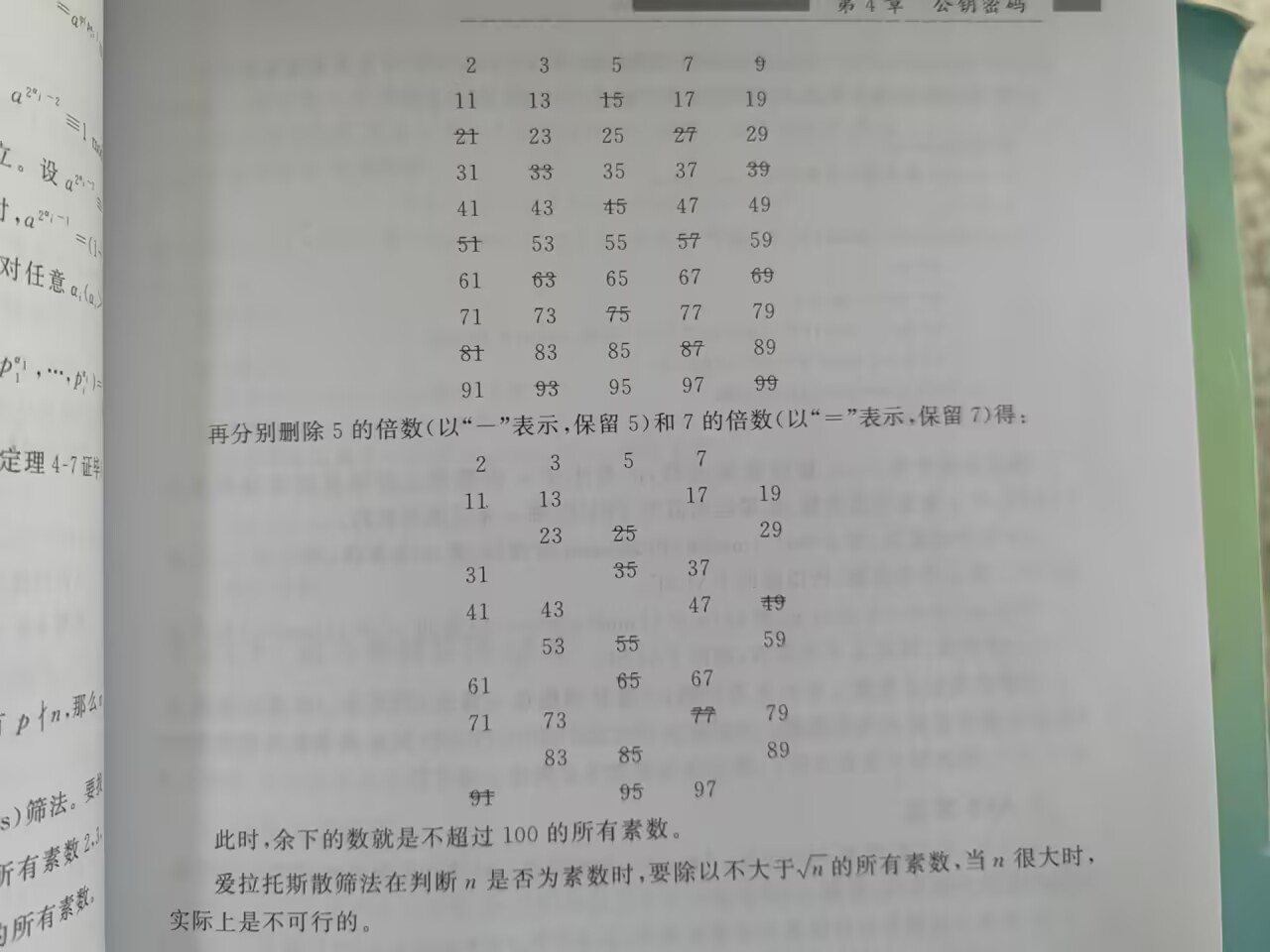
1. **实验内容**

算法理解：研究埃拉托斯特尼筛法的基本思想，明确其实现步骤。

编程实现：创建一个布尔数组，用于标记每个数是否为素数。遍历数组，根据算法规则标记非素数。收集并输出所有素数。

结果验证：通过对比已知的小范围素数列表，验证程序的正确性。





1. **实验过程**

1.算法原理回顾

埃拉托斯特尼筛法是一种高效的求素数算法，其基本思想是从2开始，将每个素数的各个倍数标记为非素数。具体步骤如下：

创建一个布尔数组isPrime，长度为n+1（本题中n=10000），初始值全为true，表示假设所有数都是素数。将isPrime[0]和isPrime[1]设为false，因为0和1不是素数。从2开始遍历到 n ，对于每个数i，如果isPrime[i]为true，则将其所有倍数（从i 2 开始，到n为止，步长为i）标记为false。遍历结束后，isPrime数组中值为true的下标即为素数。

2.Java实现步骤

int n = 10000;

boolean prime[] = new boolean[n + 1];

// 初始化所有数为素数

for (int i = 2; i <= n; i++) {

prime[i] = true;

}

// 实现埃拉托斯特尼筛法

for (int p = 2; p \* p <= n; p++) {

if (prime[p]) {

// 将p的所有倍数标记为非素数

for (int i = p \* p; i <= n; i += p) {

prime[i] = false;

}

}

}

// 输出所有素数

System.out.println("10000以内的素数有:");

for (int i = 2; i <= n; i++) {

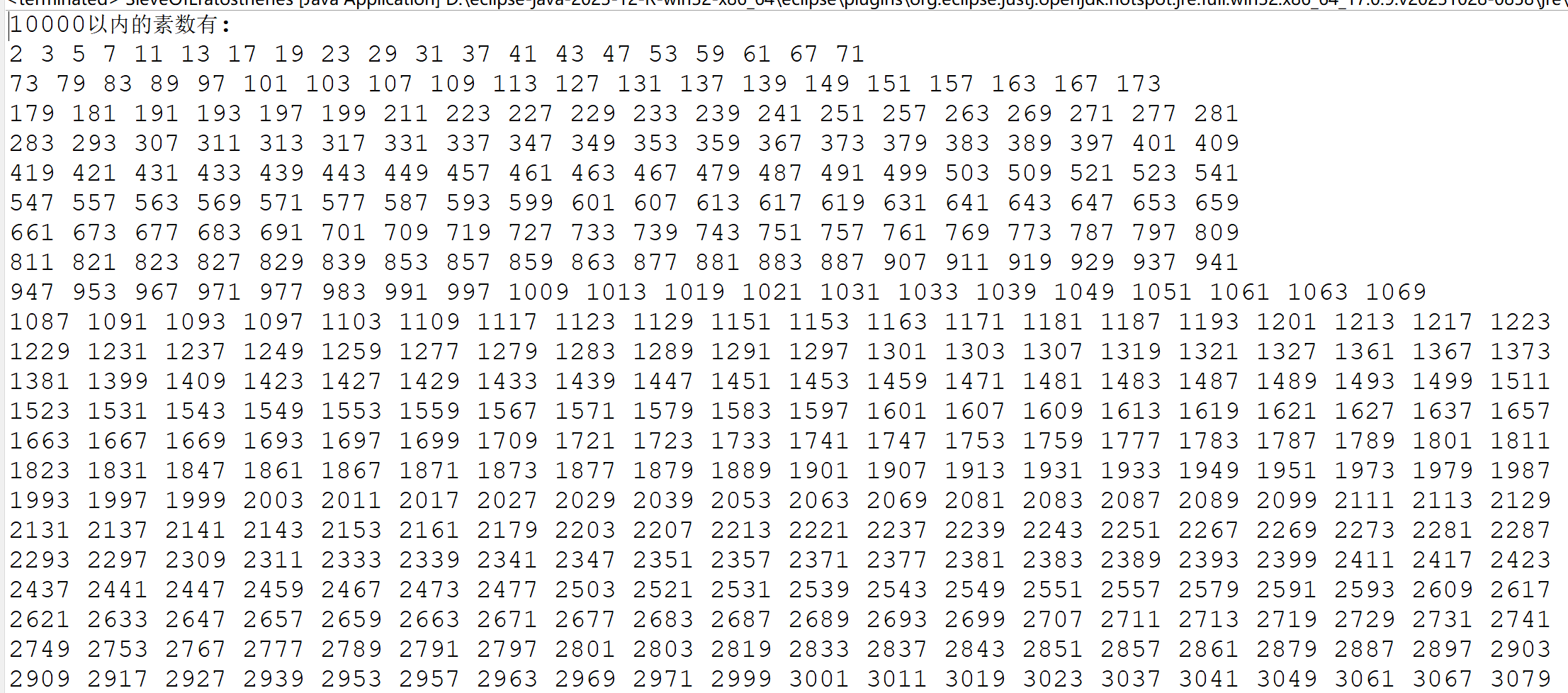
if (prime[i]) {

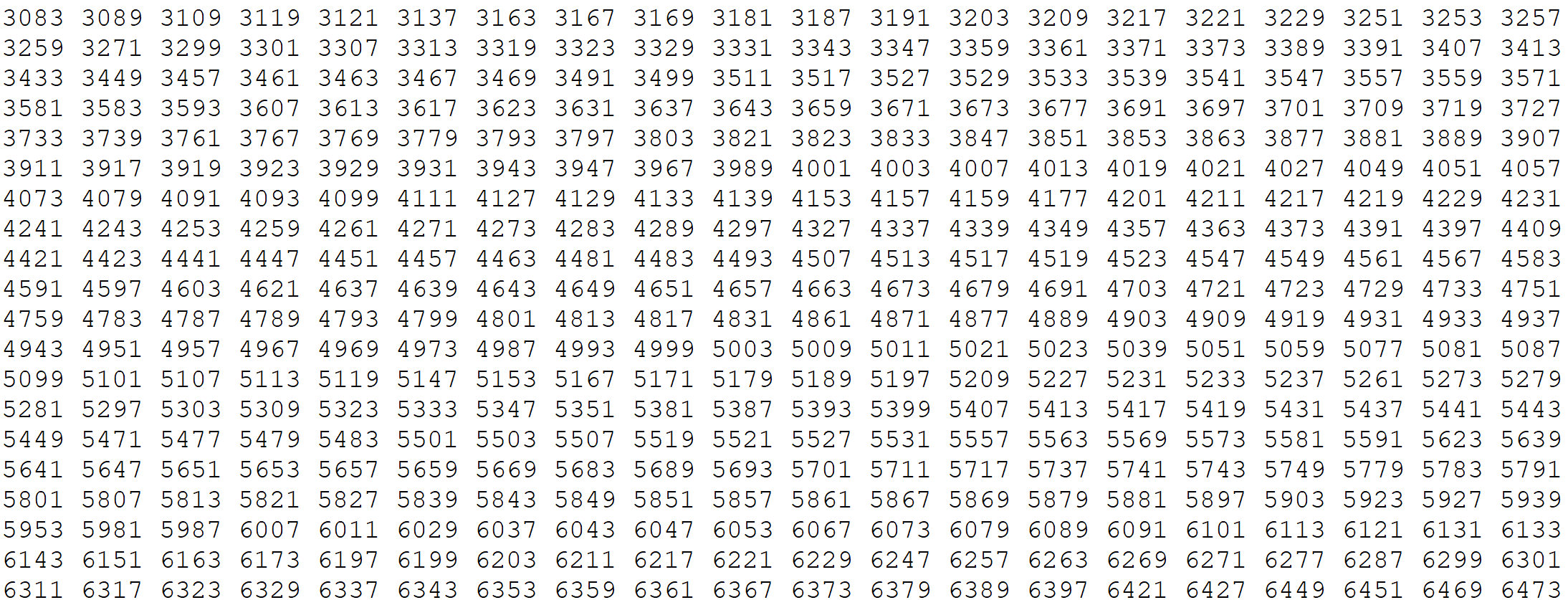
System.out.print(i + " ");

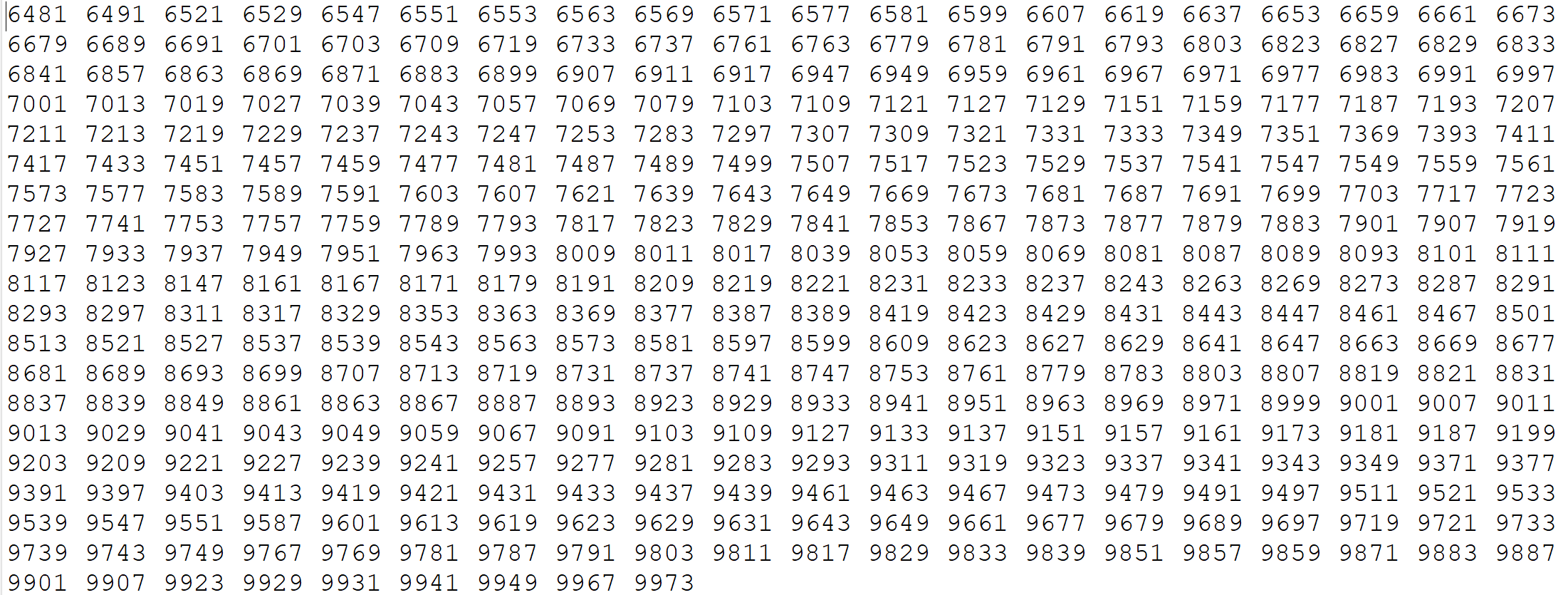
}

}

1. **实验结果**







1. **实验总结**

算法效率：

时间复杂度：埃拉托斯特尼筛法的时间复杂度为O(nloglogn)，在处理大规模数据时效率较高。

空间复杂度：空间复杂度为O(n)，需要额外的布尔数组来存储素性信息。

优化方向：

可以进一步优化内层循环的起始点和步长，减少不必要的计算。

对于极大的n，可以采用分段筛法（Segmented Sieve）来降低内存占用。

实验收获：

通过实验，深入理解了埃拉托斯特尼筛法的工作原理和实现细节。掌握了用Java实现算法的基本方法，提高了编程能力和算法分析能力。认识到算法选择和数据规模对程序性能的重要影响，为后续的学习和研究奠定了基础。