

## 204. Count Primes

📅 发表于 2016-03-03 | 📁 分类于 [LeetCode](#) , [Java](#) | 👁 阅读次数: 64

Total Accepted: **54299**

Total Submissions: **230350**

Difficulty: **Easy**

### Description:

Count the number of prime numbers less than a non-negative number,  $_n$ .

### Credits:

Special thanks to @mithmatt for adding this problem and creating all test cases.

### Hint:

- Let' s start with a isPrime function. To determine if a number is prime, we need to check if it is not divisible by any number less than  $_n$ . The runtime complexity of isPrime function would be  $O(_n)$  and hence counting the total prime numbers up to  $_n$  would be  $O(_n^2)$ . Could we do better?
- As we know the number must not be divisible by any number  $> _n / 2$ , we can immediately cut the total iterations half by dividing only up to  $_n / 2$ . Could we still do better?
- Let' s write down all of 12' s factors:

```
1  2 × 6 = 12
2  3 × 4 = 12
3  4 × 3 = 12
4  6 × 2 = 12
```

As you can see, calculations of  $4 \times 3$  and  $6 \times 2$  are not necessary. Therefore, we only need to consider factors up to  $\sqrt{_n}$  because, if  $_n$  is divisible by some number  $_p$ , then  $_n = _p \times _q$  and since  $_p \leq _q$ , we could derive that  $_p \leq \sqrt{_n}$ . Our total runtime has now improved to  $O(_n^{1.5})$ , which is slightly better. Is there a faster approach?

```
1  public int countPrimes(int n) {
2      int count = 0;
3      for (int i = 1; i < n; i++) {
4          if (isPrime(i)) count++;
5      }
6      return count;
7  }
8
9  private boolean isPrime(int num) {
10     if (num <= 1) return false;
11     // Loop's ending condition is i * i <= num instead of i <= sqrt(num)
12     // to avoid repeatedly calling an expensive function sqrt().
13     for (int i = 2; i * i <= num; i++) {
14         if (num % i == 0) return false;
15     }
16     return true;
17 }
```

- The Sieve of Eratosthenes is one of the most efficient ways to find all prime numbers up to  $_n$ . But don' t let that name scare you, I promise that the concept is surprisingly simple.

	2	3	4	5	6	7	8	9	10	Prime numbers
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	
41	42	43	44	45	46	47	48	49	50	
51	52	53	54	55	56	57	58	59	60	
61	62	63	64	65	66	67	68	69	70	
71	72	73	74	75	76	77	78	79	80	
81	82	83	84	85	86	87	88	89	90	
91	92	93	94	95	96	97	98	99	100	
101	102	103	104	105	106	107	108	109	110	
111	112	113	114	115	116	117	118	119	120	

Sieve of Eratosthenes: algorithm steps for primes below 121. "Sieve of Eratosthenes Animation" by Skopp is licensed under CC BY 2.0. We start off with a table of  $_n$  numbers. Let' s look at the first number, 2. We know all multiples of 2 must not be primes, so we mark them off as non-primes. Then we look at the next number, 3. Similarly, all multiples of 3 such as  $3 \times 2 = 6$ ,  $3 \times 3 = 9$ , ... must not be primes, so we mark them off as well. Now we look at the next number, 4, which was already marked off. What does this tell you? Should you mark off all multiples of 4 as well?

- 4 is not a prime because it is divisible by 2, which means all multiples of 4 must also be divisible by 2 and were already marked off. So we can skip 4 immediately and go to the next number, 5. Now, all multiples of 5 such as  $5 \times 2 = 10$ ,  $5 \times 3 = 15$ ,  $5 \times 4 = 20$ ,  $5 \times 5 = 25$ , ... can be marked off. There is a slight optimization here, we do not need to start from  $5 \times 2 = 10$ . Where should we start marking off?

- In fact, we can mark off multiples of 5 starting at  $5 \times 5 = 25$ , because  $5 \times 2 = 10$  was already marked off by multiple of 2, similarly  $5 \times 3 = 15$  was already marked off by multiple of 3. Therefore, if the current number is  $_p$ , we can always mark off multiples of  $_p$  starting at  $_p^2$ , then in increments of  $_p$ :  $_p^2 + _p$ ,  $_p^2 + 2\_p$ , ... Now what should be the terminating loop condition?

- It is easy to say that the terminating loop condition is  $_p < _n$ , which is certainly correct but not efficient. Do you still remember Hint #3?

- Yes, the terminating loop condition can be  $_p < \sqrt{_n}$ , as all non-primes  $\geq \sqrt{_n}$  must have already been marked off. When the loop terminates, all the numbers in the table that are non-marked are prime. The Sieve of Eratosthenes uses an extra  $O(_n)$  memory and its runtime complexity is  $O(_n \log \log _n)$ . For the more mathematically inclined readers, you can read more about its algorithm complexity on [Wikipedia](#).

```
public int countPrimes(int n) {
    boolean[] isPrime = new boolean[n];
    for (int i = 2; i < n; i++) {
        isPrime[i] = true;
    }
    // Loop's ending condition is i * i < n instead of i < sqrt(n)
    // to avoid repeatedly calling an expensive function sqrt().
    for (int i = 2; i * i < n; i++) {
        if (!isPrime[i]) continue;
        for (int j = i * i; j < n; j += i) {
            isPrime[j] = false;
        }
    }
    int count = 0;
    for (int i = 2; i < n; i++) {
        if (isPrime[i]) count++;
    }
    return count;
}
```

Java:

```
1  public class Solution {
2      public int countPrimes(int n) {
3          if (n < 3)
4              return 0;
5          int count = n / 2;
6          boolean[] isPrime = new boolean[n];
7          for (int i = 3; i * i < n; i += 2) {
8              if (isPrime[i])
9                  continue;
10             for (int j = i * i; j < n; j += 2 * i) {
11                 if (!isPrime[j]) {
12                     --count;
13                     isPrime[j] = true;
14                 }
15             }
16         }
17         return count;
18     }
19 }
```

### 打个小广告

欢迎加入我的小专栏「[基你太美](#)」一起学习。

打赏

本文作者：布兰柯基

本文链接： <https://blankj.com/2016/03/03/204-count-primes/>

版权声明： 本博客所有文章除特别声明外，均采用 [CC BY-NC-SA 3.0](#) 许可协议。转载请注明出处！

# Math # Hash table

< 7. Reverse Integer

125. Valid Palindrome >

Blankj's Blog

select \* from utils

🏠 首页

👤 关于

📁 分类

🔖 标签

📅 归档

🌟 我的星球

♥ 面经

🔗 我的开源库

⚡ Go 速查

📖 GitBook

👥 小伙伴们

👍 内推

🔍 搜索



布兰柯基

161

日志

38

分类

85

标签

🐙 GitHub

📖 简书

💰 掘金

📖 CSDN

🐦 微博

0 comments

Anonymous ▾



Leave a comment

[🔗 Markdown is supported](#)

Login with GitHub

Preview

Be the first person to leave a comment!