黑客派

Java，Node，Python 运行速度比较

作者：someone45057
原文链接：https://hacpai.com/article/1555564083350
来源网站：黑客派
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

<img src="https://static.hacpai.com/images/img-loading.svg" alt="" data-src="http://stat
c.simpledesktops.com/uploads/desktops/2018/08/04/six.png">
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
(adsbygoogle = window.adsbygoogle || []).push({});
</script>
<h2 id="首先声明我并不是想证明某一个语言比另外一个好-因为每一个语言都是图灵完备的">首先
明我并不是想证明某一个语言比另外一个好，因为每一个语言都是图灵完备的</h2>
撰写该博客的起因是看到朋友转发了一条这样的微博：
<img src="https://static.hacpai.c
m/images/img-loading.svg" alt="reason.png" data-src="https://img.hacpai.com/file/2019/04/
eason-82c06298.png?imageView2/2/interlace/1/format/jpg">
为了保证公平，三种语言
代码逻辑都是一致的，并且都是在同一个电脑上运行的
话不多说，直接上代码
<h3 id="Python代码-3-6-5-">Python 代码（3.6.5）</h3>
<pre><code class="highlight-chroma">import time

判断是否为质数

def isPrime(num):

for i in range(2, (int)(num / 2)):

if num % i == 0:

return False

return True

获取3出现的次数

def getThreeNumbers(num):

res = 0

for i in range(3, num, 2):

n = i

while(n not found render function for node [type=NodeHTMLEntity, Tokens=>]not fo
nd render function for node [type=NodeHTMLEntity, Tokens=>] 0):

if n % 10 == 3:

res = res + 1

n = int(n / 10)

print ('3出现的次数：' + str(res))

获取微信ID

def getWechatID(num):

for i in range(2, int(num / 2)):

```
if num % i != 0:
continue
div = int(num / i)
if !isPrime(i) or !isPrime(div):
continue
res = ''
if div not found render function for node [type=NodeHTMLEntity, Tokens=>]not found
ender function for node [type=NodeHTMLEntity, Tokens=>] i:
res = res + str(div) + str(i)
else:
res = res + str(i) + str(div)
getThreeNumbers(int(res))
print('微信ID：NY' + res)
return

start = time.time()
getWechatID(707829217)
end = time.time()
print ('Time cost:' + str((end - start)))
</code></pre>

<h3 id="Node-JavaScript-代码-10-15-3-">Node(JavaScript)代码（10.15.3）</h3>
<pre><code class="highlight-chroma">console.time('Time cost')
//判断一个数是否为质数
const isPrime = num =&gt; {
  for(let i = 2; i &lt; Math.floor(num / 2); i++){
    if(num % i == 0) return false
  }
  return true
};
//得到3出现的次数
const getThreeNumbers = num =&gt; {
  let  res = 0
  for(let i = 3; i &lt;= num; i+=2){
    let n = i
    while(n &gt; 0){
      if(n % 10 == 3) res++
      n = Math.floor(n / 10)
    }
  }
  return res
};
//得到微信ID
const getWechatID = num =&gt; {
  for(let i = 2; i &lt; Math.floor(num / 2); i++){
    if(num % i != 0) continue
    let div = num / i
```

```
    if(isPrime(i) &amp;amp;&amp;amp; isPrime(div)){
      let res = div &gt; i ? `${div}${i}` : `${i}${div}`
      console.log(`3的次数：${getThreeNumbers(res)}`);
      return res
    }
  }
}
console.log(`微信ID：NY${getWechatID(707829217)}`);
console.timeEnd('Time cost')
</code></pre>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<h3 id="Java代码-1-8-0-201-">Java 代码（1.8.0_201）</h3>
<pre><code class="highlight-chroma">public class Test {


}
</code></pre>

<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>

<!-- 黑客派PC帖子内嵌-展示 -->

<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<h3 id="运行结果">运行结果</h3>
<ul>
 <li>Python<br> <img src="https://static.hacpai.com/images/img-loading.svg" alt="image.p
g" data-src="https://img.hacpai.com/file/2019/04/image-190370c0.png?imageView2/2/interl
ce/1/format/jpg"></li>
</ul>
<p><strong>注意一下，这里的单位是秒不是毫秒（ms），转化成毫秒就是 1926298ms</strong
</p>
<ul>
 <li> <p>Node<br> <img src="https://static.hacpai.com/images/img-loading.svg" alt="ima
e.png" data-src="https://img.hacpai.com/file/2019/04/image-8c7bfab8.png?imageView2/2/in
erlace/1/format/jpg"></p> </li>
 <li> <p>Java<br> <img src="https://static.hacpai.com/images/img-loading.svg" alt="image
png" data-src="https://img.hacpai.com/file/2019/04/image-ed8d3078.png?imageView2/2/int
rlace/1/format/jpg"></p> </li>
</ul>
```

<h3 id="分析">分析</h3>
且不谈优化，我知道我的解法肯定有待优化，至少这里三种代码的逻辑是一致的
 从结果数来看，python 的确是慢了很多，Java 的确是快很多
 Java：15277ms
 Node：70327ms
 Python：1926298ms
 下面分析原因：
<h4 id="算法复杂度">算法复杂度</h4>
由于用到了双层循环，所以算法的时间复杂度是

O(n^2)$$ 其实准确的说，这里的时间复杂度是 $$O(\frac{1}{2} * \frac{1}{2} * n^2 + \frac{1}{2} * * log_{10}n)$$ 只保留最高次项，同时忽略最高项的系数得到 $$O(n^2)$$ #### 代码编写时间 在里具体的编写代码时间不好测量，因为我对每个语言的熟练程度不一样 但是java代码的行数最多，至能说明键盘敲得次数更多 #### 语言特性 * 强弱类型（动静态类型） Java是强类型语言（静态类型） 而Python，Node（JavaScript）是弱类型语言（动态类型） 这意味着Python跟Node需要在运行时类型检查以及转换，所以速度自然会受影响 * 语言类型 Java是编译型语言 Node（JavaScript）是混型语言（为啥是混合型，我也是Google了一下，感兴趣的自己Google） Python是解释型语言 Pytho需要对代码进行读取，词法分析，解析，编译，解释和运行 Node第一次也需要如上步骤，但是执行同的逻辑时，将会跳过读取，词法分析，解析，编译，解释，直接运行 Java则是先编译，然后直接运 ### 结论 * 语言本身并无优劣之分 * 当需要重复计算时，Python的速度的确慢很多，Java的确有优势 * 选择哪一种语言取决于你想用它做什么以及你的成本是多少，包括硬件成本，时间成本，只有根据的具体需求才能选择对的语言去开发，不能空谈效率 --- ## 算法优化 想了想还是继续优化一下算法，之前给出的复杂度计算有误，已经改正 ### 循环边界判断优化 感谢（@[kafuly](https://hacpai.com/member/kafuly)）给出的建议 将`isPrime()`以及`getWechatID()`函数中的循环边界都改成`i <= num / 3`，这里三种语言代码稍稍有点区别： * Java直接修改即可 `i <= num / 3` * Node 需要调Math.floor() `i <= Math.floor(num / 3)` * Python 需要调用int()函数 `i <= int(num / 3)` 优化毕后，复杂度是： $$O(\frac{1}{3} * \frac{1}{3} * n^2 + n*log_{10}n)$$ ### 进一步思考 这里我们计算一下复杂度里第一项与第二项的大小： * 第一项: $$\frac{1}{3} * \frac{1}{3} * n^2 $$ 这里n的大值是8171 所以计算次数是 $$8171 * 8171 * 1/9 \approx 7418360$$ * 第二项： $$n * log_{10}n $ 这里的n最大值是866278171，所以计算次数为： $$866278171 * log_{10}866278171 \approx 742497480$$ 这里可以看到第二项的计算次数占了很大的比重 ### 继续优化，下面分情形探讨一下n内的奇数序列中3出现的个数 #### n为10的整数次幂数时 * n=10 $$f(10) = 1$$ * n = 100 3, 13, 23 33, 43, 53, 63, 73, 83, 93 总共10次 30, 31, 32, 33, 34, 35, 36, 37, 38, 39出现10次 只保留奇数序，且33重复计算一次，所以结果为： $$f(100) = 10*1+\frac{10}{2}=15$$ * n = 1000 $$10 * 15 + \rac{100}{2} = 200$$ * 可以推出： <p style="text-align:center;font-size:20px;">f(n) = f(10&t;sup>n-1</sup>)*10 + 10ⁿ⁻¹/2</p> #### n = 10的整次幂的整数倍数时 * n = 300 $$3 * f(100) = 3 * 15 = 45$$ * n = 700 $$ 7 * f(100) + 100 / 2 = 7 * 15 + 50 = 155 $$ * 推出： 当系数<=3时 <p style="text-align:center;font-size:20px;">f(a10^b) = a*f(10^b)</p> 当系数>3时 < style="text-align:center;font-size:20px;">f(a*10^b) = a*f(10<sup gt;b</sup>) + 10ⁿ⁻¹/2</p> #### n 为一般情况时 * n = 176 <p style="font-size:20px;padding-left:8em;"> f(1764) = f(1000) + 7*f(100) + 10*f(10)/2 6*f(10) + 10/2 + f(4)</br> f(1000) = 10*f(100) + 100/2</br> f(100) = 10*f(10) + 10 / 2</br> f(10) = 1</br> f(4) = 1</br> => f(100) = 15 </br> =>f(100) = 200</br> =>f(1764) = 200 + 7*15 + 100/2 + 6*1 + 5 + 1 = 367 </p> ### 优化之后的代码 优化之后，我们的`getThreeNumbers()`方法看上去就是这样的: * Java ``` //获取3现的次数 public static int getThreeNumbers(int num){ if(num < 3){ return 0; } if(num >= 3 && num <= 10){ return 1; } //得到num比10的多少次方大（或相等） int pow = (int Math.log10(num); int afterPow = (int)Math.pow(10, pow); //得到系数 int times = num / afterPw; //得到剩余部分 int remindPart = num % (times * afterPow); if(times > 3){ return times*f(ow) + afterPow/2 + getThreeNumbers(remindPart); } return times*f(pow) + getThreeNumbers remindPart); } //获得10的整数次幂的结果 public static int f(int pow){ if(pow == 1){ return 1; } reurn 10*f(pow-1) + (int)Math.pow(10, pow - 1)/2; } ``` * Node代码 ``` //得到3出现的次数 const geThreeNumbers = num => { if(num < 3){ return 0; } if(num >= 3 && num &lt = 10){ return 1; } let pow = Math.floor(Math.log10(num)) let afterPow = Math.pow(10, pow) le times = Math.floor(num / afterPow) let remindPart = num % (times * afterPow) if(times >){ return times*f(pow) + afterPow/2 + getThreeNumbers(remindPart) } return times*f(pow) +

etThreeNumbers(remindPart) }; //获得10的整数次幂的结果 const f = pow => { if(pow == 1){ eturn 1 } return 10 * f(pow - 1) + Math.pow(10, pow-1)/2 }; ``` * Python ``` # 获取3出现的次数 d f getThreeNumbers(num): if num < 3: return 0 if num >=3 and num <= 10: return 1 p w = int(math.log10(num)) afterPow = int(math.pow(10, pow)) times = int(num / afterPow) rem ndPart = num % afterPow if times > 3: return times*f(pow) + afterPow/2 + getThreeNumb rs(remindPart) return times*f(pow) + getThreeNumbers(remindPart) # 获得10的整数次幂的结果 def f(pow): if pow == 1: return 1 return 10*f(pow - 1) + math.pow(10, pow-1)/2 ``` ### 优化 果 #### Java ![image.png](https://img.hacpai.com/file/2019/04/image-3e778eb8.png) 1ms # ## Node ![image.png](https://img.hacpai.com/file/2019/04/image-0ff98022.png) 11.6ms ### Python ![image.png](https://img.hacpai.com/file/2019/04/image-ba0c0889.png) 11.9ms ### 再次分析 可以看到，Java的速度优势已经体现不出来多少了，10ms的差距是感受不到的 反而这个时 java的编译速度+代码撰写速度比较长了 #### 再看复杂度 现在的总复杂度其实还是： $$O(n^2)$$ * 第一项的计算次数还是 $$8171 * 8171 * 1/9 \approx 7418360$$ * 第二项的计算次数 因为用了递 ，其实这里还可以优化，可以将递归改成非递归，这里就不继续写了 $$f(n)=log_{10}n + log_{10}(n 1) + ... + log_{10}1$$ 将n = 866278171代入得： $$f(n)\approx36$$ 到这里，解题+优化的过程 部结束

<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js">
</scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342" data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>