质数的来源

质数是从哪里来的？

自然数中，除了0和1之外，要么是质数，要么是合数。合数容易理解，它是质数幂次的乘积（唯一分解定律），或者说，它是质数的比较结果。这可以认为是，

也就是说，一系列质数的幂次为周期的最小单位的比较结果。可是质数本身从哪里来？当然可以认为，它是由1堆积而来的，但是这是一种过程的说法，而对于存在本身而言，需要相应的机制。

比如在自然数全加和的运算中，我们可以得出质数2以及3的来源，它们皆来自于无限的自身等于自身一部分的这种结构。那么，根据这个经验，我们可以假定其它质数也是来自于自身等于自身一部分的这种结构，或者说，来自于无限，而这里的无限体现为自我迭代的过程。

我们总是试图寻找质数公式，也就是说一个能够计算出所有质数的公式来，但这并不容易。在研究黎曼ζ函数的过程中受到了启发，引入虚数单位，用虚数单位的无限性和周期性周期地替换无穷，得到多个答案再将答案综合起来。

下面我们还是用这样一个方法，尝试推导出产生更多质数的质数公式。

考虑黎曼函数的两种等价的形式，

的特例ζ(2)以及它的两种表达方式，

求前项的和，

为了能够积分（这种数列没有求和公式，但对应的函数却有积分公式），我们将自然数先转化为无穷小量的倍数，具体来说，积分最小单位是，下限是（用来代表0），上限分别为，正好构成第一个周期（从第二象限开始），我们目前也只处理第一个周期的情况。

若计算加法，则可以得到

所以可以看出，乘法的时候，

加法的时候，

把乘法和加法的结果配合在一起，

其它情况，

以及，

以上三种情况暂时不讨论，我们看最重要的一种情况，

我们知道，

所以，

把等式写成迭代式，

令，

原方程变为，

最终得到，

求正数解，

如果修改积分下限为可变的数值，方程可以进一步化为，

用上述公式转化出的程序，可以求出2之后的3，5，7，11，这四个质数。C#程序如下，

static void TestNextPrime(int n = 4)

{

double A = 1;

double B = 1;

double m = 1;

long pk = 2;

List<long> primes = [pk];

for (int i = 0; i < n; i++)

{

double pk2 = pk \* pk;

double pk4 = (pk2 - 1) \* (pk2 - 1);

A \*= (pk4 == 0 ? 1 : pk4);

B \*= (pk2 \* pk2);

double sqrt\_B = Math.Sqrt(B);

double sqrt\_B\_A = Math.Sqrt(m \* m \* B + A);

double t1 = m \* m \* B + m \* sqrt\_B \* sqrt\_B\_A;

double t2 = m \* m \* B - m \* sqrt\_B \* sqrt\_B\_A;

double v1 = 1 + t1 / A;

double v2 = 1 + t2 / A;

double x1 = Math.Sqrt(v1);

double x2 = Math.Sqrt(v2);

double x3 = (i + 1) % 4 == 0 ? x1 - x2 : x1 + x2;

long xp = (long)Math.Round(x3);

primes.Add(xp);

m = pk;

pk = xp;

}

}

用这种求和变积分的方式，再看黎曼函数的其它形式，

也就是，

简述一下，通过引入虚数单位的倒数作为微分，将无限项和转变为可积分的函数，并对其求虚数单位的四个幂次的定积分，并将这四个结果合并为一个综合的结果。这便是这类求无限项之和函数的简易算法。

再看，

分析质数和虚数单位的关系：

如果奇质数有偶数个，

如果奇质数有奇数个

合并，

因为都是整数乘积，所以去掉的情况，认为，

所以，

由此可以知道，虚数单位是偶数。除了2和3之外的质数总数为偶数，即质数的总数是偶数。既然虚数单位是偶数，那么，

由此0必须是奇数。可是0是偶数，除非0就是下一个周期的1，或者0和下一个周期的1（几乎）重合。由此也不难看出，所谓自然数也符合，

的自己等于自己一部分的递归形式。由此自然数的个数是无限的。由于虚数单位终究是偶数，所以我们定义，

最后看ζ函数的完整形式，

仍然是求和变积分，

因为

能够使得

在0和1之间，可以合并同类项的，就只能是

自然数全加和，