我们怎么数数-哥德巴赫猜想

我们到底是怎么数数的？

问题的答案并不复杂，最基本的是一个一个的数，也可以两个两个的数，三个三个的数，总之选择了某个数量单位然后去重复它，而自然数计数的最小单位显然就是1。

我们知道自然中的许多常数，都可以表现为递归嵌套式的形式，比如，

这种形式不仅仅可以求值为一个结果，还描述了构成它的过程，所以可以认为这些特定的形式就是对应常数的数学结构。

由此可以想到，我们也可以构造自然数及其计数过程（集合和集合上的可结合运算，也就是半群）的数学结构。

具体来说，首先给出第一个自然数，

然后对其加1，

不难看出，这种写法是不区分两个不同的1的前后顺序的（类似交换群，显然也符合封闭性，可结合，单位元，但这里不涉及逆元的问题）。为了区分先后顺序，我们换用加括号的方式重新写一遍，

最内层的括号是最开始的那一个单位，它外面是第一次加法运算，次内层的括号是最开始的单位加上后来的单位的结果，它外面是第二次加法运算。如果有层的话，就是，

这种写法就像圆周率一半的展开式那样有层次性（这样就不能随意交换1的顺序了）。而这些括号，代表了包括起始的1的每个增加1的状态并链接成一个完整的过程。自然数的计数过程就是这样，所以任何自然数都一定可以写成这种以1为步进单位的递增过程。若不能写成这种过程，则不应视作一个真实存在的自然数（后面会解释）。

有了这个基础，让我们进入下一步考虑：质数是那些不能用1以外的数重复累加得到的，也就是说，质数不应写成，

这种样子。由此可见，对于一个数的计数过程，有两种方式，一种是只能一次一个的积累，例如，

还有一种，则是写成重复的递归嵌套形式，比如，

虽然它还是可以写成步进为1的递归嵌套形式，但是这种形式意味着计数单位的变化，从1变成了（比如进制）。

现在让我们考虑任意偶数，它总可以写成两个数之和（包括2）。我们选择一个是质数（假定质数包括1），另外一个数待定。根据陈氏定理，另外一个数可以是一个质数或者两个质数之积。

即任选一个偶数都只可能符合下列两种情况之一，

第一种，

第二种，

我们假定这个任意的偶数，它不能符合第一种情况，只能符合第二种情况（若发现它只能符合第二种情况不成立，则它就只能符合第一种情况。此时就证明了任选的偶数都一定可以符合第一种情况）。

具体看第二种情况，我们用，

替换的单位1，

这就构造了一种“类递归嵌套式”。

虽然说加入到括号里面并不影响计算的结果，但是从过程上要求必须每先数个1之后再数第二遍个1，一共数遍；或者先数个1，再数个1，一共数s遍。这就是我们在最开始说的加括号就保证了加1的顺序。但我们先前也说过，对一个数量计数，总可以写成单位为1的递归嵌套式，可是这里要求单位只能为或者为，且两者都大于1。

然而我们也知道，

从的视角来看，本身也是一个自然数，它就一定可以写成单位为1的递归嵌套式，这和此处它不能写成单位为1的递归嵌套式相矛盾：同一个数在相同的条件下既可以写成单位为1的递归嵌套模式又不可以写成单位为1的递归嵌套模式，是自相矛盾的（第二种形式是自相矛盾的形式），所以假设不成立。由此可知，没有一个偶数是只能写成，

而不能写成，

的形式的，也就是说，任何偶数都一定可以写成两个质数之和的形式。另一种说法，若真有这样一个偶数不能写成这种形式，它就是不存在的。

用自然语言解释一下，哥德巴赫猜想到底要说的是什么意思：一个数，之所以为数，它就是用来计东西的数量的。计数的操作又叫数数。数必须是可以一个一个 的数的，才是自然数，也就是东西的个数。若你必须一对一对的数，那其实你是把2当成了1作为单位来数的。人可以这样做，但是自然无所谓这样做，人可以把一对一对计数的结果乘以二，就得到了实际的数量。但是事物的个数，一就是一，二就是二。如果有一个数必须得一组一组的数，那它就是人的数数方式，而和实际的数量无关。所以若真的有那个数，就只能通过一个一个的数来体现和验证（也就是单位为1的递归嵌套形式），否则就没有这个数；必须3个3个的数或者8个8个的数是多余的，而缺少某个自然数也是不成立的。在哥德巴赫猜想这个例子具体来说，如果有一个偶数，它真的存在，也就是说那个偶数的数量是实际中事物的真实数量，而且一定要分成两个数的和，那么这两个数的单位就必须都是1（所以它们都是质数）。人可以用大于1的单位度量数量，但是数量本身不可能选择除了1之外的其它单位。

有了这个认识，那么我们说，所有3的倍数，都可以用3个质数相加得到；所有4的倍数都可以用4个质数相加得到，所有m的倍数都可以用m个质数相加得到。原则不变，具体证明就不必给出了。

本 证明受到哥德尔不完全定理的启发，首先将问题符号化，然后以添加括号的方式将过程维度引入考虑的范围，提升处理问题的维度，并对原来的问题实现降维打击。

至于为啥这个猜想如此的难，原因就在于，它其实不是数论问题，而是数学结构（群论）问题。先前的尝试，错看了问题，看错了方向。