好的，导师就位。各位同学，量子计算这个话题听起来是不是又酷又有点让人摸不着头脑？没关系。今天我们就把这个硬核的知识点，像剥洋葱一样，一层一层地解开。下面的文档是一位专家的讲解，我会带着大家一起，把它从“天书”变成我们都能听懂的大白话。让我们开始吧。

【原文】
很多科普媒体在介绍量子计算时会给出一种特定的总结，
我几乎可以保证这种总结会导致误解。
这个总结大概是这样的。
在经典计算机中，数据是用比特存储的，也就是一串0和1的序列，
但在量子计算机中，你可以将某个固定长度的所有可能的比特序列
同时表示在一个被称为“叠加态”的东西里。
有时，这些总结所暗示的意思是，量子计算机之所以更快，
基本上是因为它能做经典计算机所做的一切，
只是对所有这些序列并行处理。
现在，这确实触及了一些真实的东西，
但让我试着向你证明为什么我认为这会导致误解，
我将用一个小测验来证明。
为了设置场景，我想让你想象我有一个神秘函数，
我告诉你，在从0到n-1的所有数字中，有一个特定的秘密数字，
如果你把这个值输入到我的函数中，它会返回真，
但如果你输入任何其他值，它会返回假。
并且，假设你不能查看函数的内部来了解任何信息，
你唯一能做的就是用数字去尝试它。
热身问题是，平均来说，
你需要调用这个神秘函数多少次才能找到那个密钥？
嗯，如果这个场景是在一台普通的经典计算机上，
你真的没有比猜测和检查更好的办法了。
你遍历所有数字，也许你运气好能早点找到，
也许你运气不好直到后面才出现，但平均来说，
对于一个有n种可能性的列表，找到密钥需要n的一半次尝试。
现在，在计算机科学中，人们关心运行时间的规模变化。
如果列表大小是原来的十倍，需要的时间会增加多少？
计算机科学家有一种对运行时间进行分类的方法。
他们会称之为 O(n)，其中那个大O表示可能
存在一些像二分之一这样的常数，或者一些增长速度慢于n的因子，
但n这个因子才解释了随着n的增长，运行时间扩展得有多快。

如果n增加了十倍，运行时间也会增加十倍。
好了，这是你的小测验。
对于同样的情景，但在一台量子计算机上，
找到密钥的最佳运行时间是多少?
多年来，在我讲授的某个讲座中，我多次问过这个测验的一个变体，
我通常提供的选项是 O(√n)、
O(log n)、O(log(log n)) 和 O(1)，
这里的O(1)意味着运行时间只是某个常数，
它实际上不随n的增长而增长。
现在，公平地说，我还没有定义量子计算。
事实上，从头开始定义它将是这个视频的目标。
所以，在没有向你展示这个神秘函数在这种情景下会是什么样之前，
这个问题有点不着边际。
但这并不是一个我要给你打分的测验之类的东西，

【解读】
各位同学，请看第一段。作者开篇就指出了一个我们经常在科幻电影或科普文章里看到的观点：量子计算机的超能力在于“并行计算”。什么意思呢？就是说，如果一台普通电脑一次只能试一把钥匙开锁，那么量子计算机就像拥有无数个“分身”，可以同时把成千上万把钥匙都试一遍，瞬间找到正确的那一把。这个比喻听起来非常强大，对吧？作者说，这个比喻虽然沾了点边，但它具有极大的误导性。为了证明这一点，他设计了一个思想实验。

这个实验是这样的：想象一个“黑箱函数”，你不知道它内部的原理，但你知道在0到n-1这么多个数字里，只有一个“天选之子”输入进去，函数会输出“真”（True），其他所有数字输进去都输出“假”（False）。你的任务就是找到这个“天选之子”。

我们先用熟悉的经典计算机来思考。如果n等于一百万，你要找这个秘密数字，能怎么办？最朴素的方法就是“暴力破解”：从0开始，一个一个试。0，不对；1，不对；2，不对……最坏的情况，你可能要试到最后一个才找到。最好的情况，第一次就蒙对了。平均下来，大概要试n/2次，也就是五十万次。

这里作者引入了一个非常重要的概念，叫做“大O表示法”（Big O notation），比如O(n)。大家不要被这个符号吓到，它其实是个“懒人工具”，用来描述算法效率的“增长趋势”。O(n)的意思是，算法的运行时间（或者说尝试次数）与问题规模n成“正比关系”。比如，你的搜索范围从一百万（n）扩大到一千万（10n），那么你平均要试的次数也差不多会变成原来的十倍。前面的那个1/2的常数在大O表示法里被忽略了，因为它不影响整体的增长趋势。

铺垫完毕，现在作者抛出了真正的问题：如果用一台量子计算机来解决同样的“大海捞针”问题，它的效率，也就是它的大O，会是什么？他给了几个选项：O(√n)、O(log n)、O(1)。这

些选项代表了不同级别的“加速”。O(1)意味着“瞬间完成”，不管n有多大，时间都一样。O(log n)是“指数级加速”，我们熟悉的二分查找法就是这个效率，n增大很多，时间只增加一点点。O(√n)则是一种介于两者之间的加速。作者说，这个问题不是为了考我们，而是为了探测我们的“直觉”——一个很可能被“并行计算”这个误导性比喻所塑造的直觉。

【原文】
这只是在我们深入之前对直觉的一次检验。
原则上，这是同一个任务。
好比大海捞针，你想要在众多选项中发现
哪一个值能唯一地触发某个函数。
上个月我把它作为一个 YouTube 帖子发布了，有十万热心的观众作了回答。
最近一次我现场提问的对象是一群斯坦福的学生。
我也向参加国际数学奥林匹克竞赛的人们提出了这个问题。
在所有这些以及许多其他场合，答案的分布都非常相似。
最常见的答案总是 O(1)。
而这是错的，而且我非常确定这源于那个误导性的总结。
那个总结暗示，你会把所有需要搜索的 n 个值
放入一个神秘的叠加态中，
然后并行处理它们，
然后答案就会以某种方式揭示出来。
第二常见的答案通常是 O(log n)，而这也是错的。
你会称之为指数级加速。
例如，如果你将列表的大小增加 10 倍，
一个 O(log n) 的运行时间每次只会增加一个相同的固定增量。
我怀疑这个错误的答案源于一个误解，
即认为量子计算机普遍要强大得多。
在某些非常特殊的问题上，你确实可以实现指数级加速。
最著名的例子可能就是用于分解大数的 Shor 算法。
但大多数问题并非如此。
在这种情况下，正确答案是 O(√n)。
这更能代表你通过量子计算机所能获得的典型加速效果。
1994 年，有人证明了量子计算机在这项任务上
不可能做到比 O(√n) 更好。
然后两年后，Lov Grover 找到了一个具体的
算法，确实达到了那个运行时间。
所以搜索一个有一百万个选项的集合，大约需要一千步。
一个有一万亿个选项的集合，大约需要一百万步。

这个大 O 符号实际上隐藏了一些很有趣的东西，
那就是在精确的运行时间里有一个 π/4 的常数，
而这个 π 背后还有一个非常有趣的故事，我稍后会讲到。
你可能会觉得这个由一个特定值触发一个神秘函数的谜题
是经过刻意设计的。
但我希望你记住，这其实是一个通用模型的代表，
它代表了任何你知道如何快速验证解，
却不知道如何首先找到那个解的问题。
这描述了计算机科学中一大类被称为 NP 的问题。
所以，虽然平方根级的加速坦白说不如指数级加速
那样惊天动地，并且虽然大 O 运行时间通常远不如其他
实际考量重要，但像 Grover 算法这样的东西竟然是可能存在的，
本身就足以引人深思，它为加速任何 NP 问题提供了一种通用的方法。

【解读】
好了同学们，现在是揭晓答案的时刻。作者说，无论是普通网友，还是斯坦福和国际奥数竞赛的顶尖学生，绝大多数人都选了O(1)或O(log n)。然而，这两个都是错误的答案。

为什么大家会选O(1)？这恰恰证明了第一段提到的那个“并行计算”比喻的误导性有多强。人们想，既然量子计算机能“同时”检查所有n个数字，那不就等于一步到位，瞬间找到答案了吗？所以运行时间应该是个常数，不随n变化，也就是O(1)。但作者明确指出，这是错的。量子计算的核心难点，恰恰在于你虽然能把所有可能性“叠加”在一起，但如何从中“提取”出你想要的那个正确答案，这绝不是一个简单的“看一眼”就能完成的过程。

那为什么会选O(log n)呢？这代表了“指数级加速”，是计算机科学中最高效的加速类型。大家可能听说过量子计算机在破解密码方面非常强大，比如著名的“Shor算法”分解大质数，它确实能实现指数级加速。所以很多人会形成一种印象：量子计算机就是无所不能的“超级神器”，能把所有难题都降维打击。但事实是，只有非常特定结构的问题才能被量子计算机指数级加速。我们这个“大海捞针”式的搜索问题，并不在此列。

正确答案是O(√n)。这被称为“平方根加速”或“二次加速”。这个结果由一位名叫Lov Grover的科学家在1996年提出的算法（Grover算法）所实现。这个加速有多厉害？我们来具象化一下。对于经典计算机的O(n)，搜索一百万个选项需要约五十万步。而对于量子计算机的O(√n)，只需要约√1,000,000 = 1000步！如果搜索范围扩大到一万亿（10^{12}），经典计算机需要五十千万亿步，而量子计算机只需要约一百万（√10^{12} = 10^6）步。差距是天壤之别。这虽然不如指数加速那么“神”，但已经是一个巨大的飞跃了。

更重要的是，作者强调，这个“大海捞针”问题不是一个特意刁难人的小谜题。它代表了计算机科学中一大类被称为“NP”的问题。NP问题有一个共同特点：验证一个答案很容易，但找到这个答案却非常困难。我举个例子：给你一个数独题的完整答案，你几分钟就能验证它对不对（验证容易）；但给你一个空白的数独盘，让你把它解出来，可能要花上几小时甚至更久（寻找困难）。破解密码、物流路径规划、药物分子设计等许多现实世界中的核心难题都属于NP问题。Grover算法的伟大之处在于，它提供了一种“通用工具”，能为所有这类“难于寻找，易于验证”的问题提供平方根级别的加速。这足以改变许多领域的游戏规则。

【原文】
我这节课的目标是逐步为大家讲解那个算法的工作原理。
它实际上非常几何化，也非常优美。
但为了理解它，我们需要先建立很多背景知识。
视频的前三分之二左右会用来建立量子计算的基础知识，我们不会用一系列类比——正如我们所见，类比会引起误解——而是把它当作一门数学来讲解，我认为这会为你提供一副眼镜，让你能更真实地看清整个领域的样貌。
这个话题有些前提在刚开始会让人觉得有点奇怪，我得提醒你，你需要花点时间来适应。
我目前的计划是，在这节课后，再出一节课讲解其背后的一些物理学原理，希望能帮助大家理解你在这里将看到的一些奇怪规则的由来。
但今天的目标是，提供一条最精简的可行路径，让大家见识一个真正、地道的量子算法。
让我再次展示经典计算和量子计算之间的对比，看看我们能否建立一个更具代表性的心智模型。
当然，经典计算机中的数据确实表现为一系列的 1 和 0，在更高的抽象层，它可能代表一种实际的数据类型，比如整数或文本；而在更低的抽象层，这些 1 和 0 代表物理世界中的某种实际事物，比如电容器两端的电压等。
当我们讨论量子计算时，这些相同的抽象层也提供了一个非常有用的框架。
在量子计算中，同样存在底层的物理测量，同样地，你用一串 1 和 0 来表示测量结果，而这串数字也可能实现你所关心的某种实际数据类型，比如一个数字。
顺便一提，我展示的这个符号叫做 ket。
几分钟后我会详细解释它，但现在你只需把它看作是用于表示某个事物来自量子计算机的符号。
让我们从中间抽象层的视角来开始描述量子计算，这意味着我们暂时将所有底层的物理学知识搁置一旁，这有点像在不讨论硬件的情况下教授计算机科学。
在经典计算机中，我们无需区分内存的状态和从内存中读出的内容，它们看起来都是相同的比特序列。但在量子计算机中，情况则完全不同。
我们今天的主要任务是理解一个叫做“态矢量”的东西，它是连续的，是计算机实际操作的对象。但它与你实际读出的值，也就是那些离散的比特序列，有着非常不寻常的关系。
在我定义这个态矢量之前，你需要知道另一个关键与经典计算机的不同之处在于，你读出的

这个值，
它看起来同样只是一串由一和零组成的序列，是随机的。
或者说得更准确一点，我应该说它通常是随机的。
你可以这样来理解：当你在量子计算机上运行一个程序时，
该程序不一定会决定一个特定的输出，
而是会决定一个涵盖所有可能输出的概率分布。

【解读】
在这一部分，作者开始为我们搭建理解Grover算法所需的知识框架。他首先强调了一个重要的学习方法：放弃那些容易产生误解的类比，转而用数学的视角来构建一个更精确、更真实的“心智模型”。这就像学习物理，我们不能总说“电流像水流”，到了一定深度，就必须用电场、电压这些更精确的数学和物理概念来思考。

接着，作者为我们描绘了经典计算机和量子计算机的层次结构，这个结构对我们理解很有帮助。无论是哪种计算机，都可以分为三层：

1. **物理底层**：经典计算机是晶体管的开关状态（高/低电压），量子计算机则是某个量子系统（如电子的自旋方向）。
2. **信息中层**：物理状态被抽象成信息的基本单位。经典计算机是“比特”（bit），明确的0或1。量子计算机则是“量子比特”（qubit），我们很快会学到它更复杂。
3. **应用高层**：这些0和1的序列被组合起来，代表我们能理解的数据，比如数字、文字、图像等。

作者指出，我们将从“信息中层”开始学习，暂时不深入探讨底层的物理实现，就像我们学编程时不需要先成为一个硬件工程师一样。

然后，本文最核心、也是最颠覆我们常识的一个概念出现了：**在量子计算机中，计算机内部的“状态”和我们最终“读出”的结果，是两码事！**

在经典计算机里，这两者是统一的。内存里存储的是0101，你读出来的就是0101，非常直接。但在量子世界，这就完全不同了。

计算机真正在操作的东西，叫做“态矢量”（State Vector）。大家可以把它想象成一个指向多维空间中某个方向的箭头。这个箭头是“连续”的，它可以指向任何方向，包含了所有可能结果的丰富信息。这就是量子叠加态的数学化身。

然而，当我们试图“看”一眼这个状态，也就是进行“测量”时，神奇的事情发生了。这个丰富的、连续的“态矢量”会瞬间“坍缩”，随机地变成一个确定的、离散的0和1序列（比如0101）。你永远无法直接看到那个“态矢量”本身，你只能看到它坍缩后的“尸体”——一个普通的经典比特串。

更关键的是，这个坍缩的结果是**概率性**的。一个量子程序运行完毕，它不会像经典程序那样给你一个唯一的、确定的答案。相反，它会给你一个关于所有可能输出的“概率分布”。比如，它可能会告诉你：你有70%的概率测量得到结果A，20%的概率得到结果B，10%的概率得到结果C。

所以，量子算法设计的精髓就在于：通过一系列精妙的操作，去操纵那个我们看不见的“态矢量”，让它最终指向一个特殊的方向。这个方向要保证当我们进行“测量”时，那个正确的答案（比如我们想找的秘密数字）被测量出来的概率会变得非常非常高，接近100%，而所有错误答案的概率都变得微乎其微。量子计算的威力，就蕴含在这场与概率的舞蹈之中。好的，同学们，大家好！今天我们要一起深入探讨一个非常前沿和激动人心的话题——量子计算。这听起来可能有点深奥，但别担心，我会像你们的学长一样，用最通俗易懂的方式，带大家揭开它神秘的面纱。我们将要解读的这份文档，是一位专家的讲解稿，它能帮助我们从一个非常巧妙的抽象层面来理解量子计算机的核心原理。让我们开始吧！

【原文】
因此，对于我在屏幕上展示的例子，这会是一台非常小的量子
计算机，你读出的内容有四个比特，
这意味着有 2 的 4 次方，即 16 种可能的输出，
而你运行的特定程序决定了一种跨越
所有这些可能输出的分布。
有些程序可能会设法将更多的概率集中在其中一个输出上，
但其他程序可能会给出在所有输出上更均匀的分布。
顺便说一下，这个你读出内容有四个比特的例子，
可以被称为一台 4 量子比特的量子计算机，而更一般地，
如果你有一台 k 量子比特的量子计算机，那就意味着有 2 的 k 次方个不同的
可能输出，任何程序都会在所有这些输出上给出一个分布，
并且你读出的内容有 k 个不同的比特。
顺便一提，“量子比特”这个词，是我
稍后会更精确地定义的另一个东西。
我确实想强调，这个分布是隐式的。
你永远不会直接看到它，你只能根据
你运行的程序来推断它是什么样子。
你永远不会以某种方式同时看到所有比特串共存。
你只会看到其中一个，它是根据这个分布随机抽取的。
在更低的抽象层上,我所描述的从内存中读取,看起来
像一次物理测量,而其随机性源于量子力学定律。

如果你对物理学,即那个更低的抽象层感到好奇，
那正是下一个视频要讲的内容。

【解读】
好的，同学们，我们来看第一段。这段话的核心思想是想告诉我们，量子计算机的输出和我们熟悉的经典计算机有何不同。

首先，请想象一下你平时用的电脑。它的基本单位是“比特”(bit)，一个比特要么是0，要么是1，非常确定。如果你有4个比特，它们就是一个确定的组合，比如 `1011` 。但量子计算机不一样，它的基本单位叫“量子比特”(qubit)。原文用一个4量子比特的计算机举例，它对应的输出不是一个确定的4位数字，而是**所有**可能的4位数字（从 `0000` 到 `1111` ，总共 $2^4 = 16$ 种）的**概率分布**。

这是什么意思呢？我们可以打个比方。经典计算机就像一个开关，要么开，要么关。而量子计算机就像一个“概率罗盘”，它的指针可以指向任何方向。当你运行一个量子程序时，你并没有直接设定结果，而是在精心**设计这个罗盘**，让它指向某些结果的概率变大，指向另一些结果的概率变小。比如，一个程序可能会让罗盘有70%的概率指向 `1011` ，10%的概率指向 `0000` ，剩下的20%均匀分给其他14个结果。

最关键的一点，原文强调了这个概率分布是“隐式”的。你永远无法像看一张饼图一样“看到”这个完整的概率分布。当你进行“读取”或者说“测量”操作时，这个概率罗盘就会瞬间“坍缩”，随机地给出一个结果。就像你转动一个精心设计过的抽奖轮盘，你只能看到它停下来的那一个结果，但轮盘本身的设计（哪个格子大，哪个格子小）决定了你抽到每个结果的概率。你运行一次程序，得到一个随机结果，比如 `1011` 。为了推断出那个隐藏的概率分布，你可能需要多次运行同一个程序，然后统计每个结果出现的频率。这就是量子计算的第一个“反直觉”但又至关重要的地方：它的过程是概率性的，而我们只能通过采样来窥探其内在规律。

【原文】
在当前这个抽象层，你只需考虑关于
所有可能比特串的概率分布。
这里还有一个有趣的规则，它确实是从底层的量子力学中浮现出来的，
那就是当你从内存中读出并看到某个特定的值之后，
计算机的底层状态会发生改变，使得现在所有的
概率都集中在你读出的那个值上。
所以，如果你反复地从内存中读取，
你会一直看到相同的值。
你可以把这些程序想象成在创建一个非常精巧和
敏感的概率分布，当你观察它的那一刻，
即从该分布中采样时，整个系统会坍缩到一个值。

现在你可能会想，这个分布是从哪里来的?
这既是最重要也是最令人困惑的部分。
你可以将计算机的状态看作是由一个大向量来描述的。
现在当我说向量这个词时，你可以就把它想成是一大串数字，
尽管你稍后会看到，把它想成作为一个在某个超高维空间中的方向。
这个向量的每个分量都对应着一个你可能
读出的值，也就是那些唯一的比特串之一。
所以在这个你读出4个比特的例子中，
状态向量会有16个不同的分量。
状态向量和所有可能输出的概率分布
不是一回事，但它们密切相关。

【解读】
这一段解释了两个非常核心的概念：**测量的坍缩**和**状态向量**。

首先，我们来谈谈那个“有趣的规则”，也就是“坍缩”。还记得我们刚才说的“概率罗盘”吗？在你不去“看”（测量）它的时候，它包含了指向所有16个结果的可能性。但是，一旦你进行测量，比如你得到了 `0011` 这个结果，整个系统就会发生巨变。那个复杂的概率分布瞬间“坍缩”了，所有的概率都100%集中到了你刚刚看到的 `0011` 上。这意味着，如果你马上再测一次，你得到的还会是 `0011` ，再测一次，还是它。这个“看一眼就定型”的特性，是量子世界的一个基本规则。你可以把它想象成一个薛定谔的猫的盒子，在你打开之前，猫是“死与活的叠加态”（一个概率分布），一旦你打开盒子看到了活猫，那它就确定是活的了，之后无论你看多少次，它都是活的。

那么，这个精巧的概率分布到底是怎么来的呢？这里，作者引入了更深层的概念——**状态向量**。同学们，你们在数学课上学过向量，比如在二维平面上，一个向量 `(3, 4)` 可以表示一个从原点出发的箭头。在量子计算里，这个概念被极大地扩展了。对于一个4量子比特的系统，它有16种可能的输出，那么它的状态就由一个有16个分量的“超级向量”来描述！你可以把它想象成一个在16维空间里的箭头。这个向量的每一个分量，比如第1个分量、第2个分量……第16个分量，分别对应着 `0000` 、 `0001` ……一直到 `1111` 这16个可能的输出。

最关键的是，原文指出了一个微妙但重要的区别：**状态向量本身不是概率分布**，但它们密切相关。这就好比一个人的基因（状态向量）和他最终可能表现出的各种特征（概率分布）之间的关系。基因决定了特征，但它本身不是特征。那么，它们具体是什么关系呢？下一段就会揭晓这个秘密。

【原文】
基本法则——我承认一开始会显得很奇怪——
是如果你取那个状态向量中每个分量的幅值并将其

平方，就能得到观测到相应输出，
即相应比特串的概率。
我得先直说，很多学习
量子计算的人都觉得这个状态向量有点奇怪。
它到底是什么？我们又为什么要把数值平方来得到概率？
就目前而言，为了简单起见，有一个重要的
细节我将略过，直到视频最后再讲。
我只是想提醒一下，对大多数人来说，这需要一点时间来适应。
为了非常清楚地说明我这里所说的基本法则，
我们假设，在这个程序处理完这个向量后，
它与某个特定比特串（比如 0011）相关联的分量
恰好是 0.5。
那么当你将该值平方，0.5的平方是0.25，
所以其可观测到的结果是，当你从内存中读取时，
你有25%的几率看到那个比特串，0011。
我要强调的一点是，这个状态向量中的值
完全可以是负数，起初你可能会认为这没有实际影响，
因为改变符号不会改变其平方值，
因此所有概率都保持不变。
概率保持不变确实如此，
但我们绝对认为这是一个不同的状态，并且正如你将看到的，
改变符号这个概念在Grover算法中扮演着非常核心的角色。

【解读】
好了，同学们，准备好迎接量子计算中最核心、也最“奇怪”的法则。这一段揭示了如何从“状态向量”得到我们之前讨论的“概率分布”。

这个法则就是：**概率 = 幅值的平方**。这里的“幅值”基本上就是状态向量里每个分量的数值。我们来拆解一下。还记得那个16维的“超级向量”吗？它的每一个分量都对应一个可能的输出结果。比如，假设对应 `0011` 这个结果的分量值是0.5。根据这个法则，我们想知道测量时看到 `0011` 的概率是多少，只需要把0.5这个值平方一下，也就是 0.5 * 0.5 = 0.25。这意味着，你有25%的概率得到 `0011` 这个结果。就是这么简单直接，但又令人费解。为什么是平方？这并非凭空捏造，而是源于量子力学的基本公设，物理学家通过无数实验验证了它的正确性。我们在这里只需要先接受这个规则。

接下来，作者提到了一个更深奥、但至关重要的一点：**状态向量的分量可以是负数**。这可能让你更困惑了。如果分量是 -0.5，它的平方 $(-0.5)^2$ 仍然是0.25，这和分量是+0.5得到的概率完全一样。那这个负号有什么用呢？这正是量子计算的威力所在。你可以把这些分量想象成

波。在物理学中，波有波峰（正）和波谷（负）。当两个波相遇时，如果都是波峰，它们会叠加变得更高（相长干涉）；如果一个波峰遇到一个波谷，它们可能会相互抵消（相消干涉）。量子计算正是利用了这种“相消干涉”的特性。通过精巧的算法设计，我们可以让那些我们不想要的错误答案对应的分量（波）相互抵消掉，同时让我们想要的正确答案对应的分量（波）相互增强。这就是为什么一个负号虽然不改变单个概率，但却能从根本上改变整个计算过程和最终结果。Grover搜索算法就是利用这个原理的绝佳范例，它能极大地提升搜索效率。

【原文】
这里，这个四量子比特的例子在屏幕上内容有点多，
而且没有太多可视化来辅助，
所以让我们把规模缩小到最小可能的情况：计算机
只有两种可能的输出，用0和1表示。
在这个最简单的情况下，状态向量只有二维，
所以我们实际上可以把它几何地表示为二维空间中的一个箭头。
在这种情况下，x坐标对应结果0，
即该坐标的平方告诉你从计算机
读出时，读到0的概率。
也许我加 Küçük bir şerit来显示这个概率会更有帮助。
你会注意到，向量越是趋向水平方向，
概率质量就越是集中在0上。
同样，y坐标也以相同的方式对应1。
一个更垂直的状态向量意味着你更有可能
在从计算机读出时看到1。
现在，请注意，因为这两个概率之和毕竟应该等于1，某事将要发生，x的平方加上y的平方应该等于1。
从几何上讲，这意味着状态向量的长度为1，
所以你可以把它想象成被限制在一个单位圆上。
更一般地说，量子计算机的状态向量的长度永远为1，
你可以把它想象成存在于某个非常高维的单位球面上。
这个二维的例子有一个特殊的名字，我已经提到过了。
它被称为qubit，是量子比特（quantum bit）的简称。
它与经典比特的相似之处在于，当你从计算机中读取时，
你要么看到0，要么看到1，但除此之外，它完全是另一种东西。

【解读】
感觉前面16维的向量太抽象了？没关系，作者非常贴心地把问题简化到了我们最熟悉的情况——二维平面。这能帮助我们真正“看到”一个量子比特是什么样的。

我们来思考最简单的量子计算机，它只有一个量子比特（qubit）。这意味着它的输出只有两种可能：0或1。按照我们之前的逻辑，它的状态向量就应该有2个分量。一个二维向量！这不正是我们在数学课上画的xy坐标系里的箭头吗？这就好理解多了。

现在，我们把这个几何图像和量子法则对应起来：

1. **x轴** 代表测量结果为 **0**。向量的 **x坐标** 的平方，就等于你测量到0的概率。
2. **y轴** 代表测量结果为 **1**。向量的 **y坐标** 的平方，就等于你测量到1的概率。

想象一个箭头从原点出发。如果它完全指向x轴正方向，比如向量是 (1, 0)，那么测量到0的概率是 $1^2=1$ (100%)，测量到1的概率是 $0^2=0$ (0%)。反之，如果它指向y轴正方向，向量是 (0, 1)，那你100%会测量到1。

最有意思的是当这个箭头处在中间位置时，比如指向45度方向。这时它的坐标大约是 (0.707, 0.707)。那么测量到0的概率就是 $(0.707)^2 \approx 0.5$ (50%)，测量到1的概率也是 $(0.707)^2 \approx 0.5$ (50%)。这就实现了0和1的“叠加态”。

这里还有一个关键的约束条件。因为测量结果要么是0，要么是1，所以这两个概率加起来必须等于100%，也就是1。用公式表达就是：$P(0) + P(1) = x^2 + y^2 = 1$。同学们，看到这个方程是不是很眼熟？这正是平面几何中**单位圆**的方程！这意味着，描述一个量子比特的状态向量，它的箭头末端必须始终位于一个半径为1的圆上。这个向量可以在这个圆上指向任何方向，而它的方向就唯一地决定了测量到0和1的概率。这就是一个量子比特的几何图像——一个在单位圆上旋转的指针。

【原文】
在数学上，一个量子比特是二维空间中的一个单位向量，
它还带有一个坐标系，其中这两个相互垂直的x和y方向
对应于你在测量时可能读出的两个值。
你应该知道，这里还有一点我暂且不谈的复杂性，
但这已经掌握了90%的正确思想。
另外，还有一个奇怪的规则，当你测量量子比特时，
看到0或1的同时，该向量会坍缩到对应的方向上。
所以，除非采取措施将该量子比特重新制备到对角线方向，
否则你之后进行的任何观测都将显示相同的结果。
很可能此时你正在想，好吧，Grant，
你要求我接受的这套前提实在是太离奇了，而如果是这样的话，
不止你一个人这么想。
我所描述的这些，正如你很可能已经无疑看出的，
基本上就是量子力学的公设。

在整个物理学中，有许多系统，比如电子的自旋或
光子的偏振，都具有这种特性，
即测量的结果是随机的，
而我们最好的物理定律让我们用一个向量来模拟该系统的状态，
就像我在这里描述的这个一样，其向量各分量大小的平方
给出了看到各种可能结果的概率。
这其实有一个特殊的名字，叫做玻恩定则。
量子比特这个概念，基本上就是对许多类似这样的可能系统的一种抽象，
就像比特是对许多可以在两个方向之一切换的物理系统的一种抽象一样。

【解读】
这一段是对前面内容的总结，并把我们从计算机科学的抽象模型拉回到了它所根植的物理现实。

首先，作者用数学语言精准地概括了量子比特（qubit）的定义：它是一个二维空间中的**单位向量**（长度为1的向量）。这个定义里包含了我们刚才讨论的所有信息：二维空间意味着两个可能的结果（0和1），单位长度保证了总概率为1。同时，作者再次强调了“坍缩”规则：一旦你测量这个向量，比如得到了结果“0”，这个向量就会瞬间“啪”地一下，倒向x轴（代表0的方向）。除非你再用某种操作去“旋转”它，否则它就会一直待在那里，你再怎么测，结果也都是0。

接下来，作者坦诚地告诉我们：你觉得这些规则很奇怪，这很正常！因为这些规则并非计算机科学家凭空发明的，它们就是**量子力学的基本公设**。也就是说，我们的微观世界，比如一个电子的自旋（可以想象成它在“向上”自旋还是“向下”自旋），或者一个光子的偏振（光波振动的方向），它们本身的行为就是这样的。它们的真实状态就是一个“状态向量”，而我们去测量它时，得到的结果是概率性的，并且遵循着一个被称为“**玻恩定则**”（Born rule）的定律——这正是我们前面说的“概率等于幅值平方”的官方名称。

最后，作者做了一个非常精辟的类比，帮助我们理解“比特”和“量子比特”的本质。经典计算机里的“比特”，是对所有具有两种稳定状态的物理系统（比如电路的开关、硬盘的磁极）的一种**抽象**。同样，“量子比特”也是一种**抽象**，它抽象了所有遵循量子力学规则的、具有两个基本状态的微观系统（如电子自旋、光子偏振等）。所以，量子计算并不是空中楼阁，它是建立在对真实物理世界深刻理解之上的一种全新的计算范式。

【原文】
顺便说一下，我一直在展示的这个符号，在任何名称中带有
“量子”一词的学科中，都用来指代这个状态空间中的单位向量。
你放在那个右矢（ket）里面的东西，通常会为该向量所代表的含义
提供某种可读的意义。

所以在我们量子比特的例子中，指向右边的单位向量通常
用一个内部包含零的右矢来表示，因为如果那是状态向量，
就意味着你确定性地从计算机中读出一个零。
同样地，垂直方向的单位向量
用一个内部包含一的右矢来表示。
而如果你继续深入阅读这方面的内容，一件你会很常我们会发现，很多人并不像我一直展示的那样用列向量来写下一个通用的量子比特，
而是喜欢把它写成这两个坐标方向上单位向量的显式加权和。
这是一种非常物理学家的惯例。
经典计算有逻辑门这个概念，
即像“与”、“或”、“非”这样的基本运算，你可以用它们来处理比特，
也可以把它们串联起来创造任意复杂的函数。
类似地，我们有量子门，
它们是可以应用于单个量子比特或多量子比特系统的某些基本操作。
它们看起来总是像以某种方式翻转或旋转状态向量。
现在我不会深入探讨所有不同量子门的细节，
但如果你好奇，我会给你看一个例子。
这是一个非常标准的门，叫做阿达马门（Hadamard gate）。

【解读】
最后这段，作者为我们介绍了量子计算中的专业“黑话”——狄拉克符号，以及进行计算的基本操作——量子门。

首先是符号。你们可能在一些科普文章里见过 `|0⟩` 和 `|1⟩` 这样的写法。这叫“右矢”（ket）符号，是物理学家狄拉克发明的，专门用来表示量子状态向量。它其实就是个“花哨的括号”，作用是给向量起个名字。

- `|0⟩` 就等同于我们前面说的指向x轴的单位向量 (1, 0)。它代表了你100%会测量到0的那个确定状态。
- `|1⟩` 就等同于指向y轴的单位向量 (0, 1)。它代表了100%会测量到1的确定状态。
  一个任意的量子比特状态，比如向量 (x, y)，就可以写成 `x|0⟩ + y|1⟩` 。这叫线性叠加，意思就是这个状态是“x份的 `|0⟩` ”和“y份的 `|1⟩` ”的混合体。这只是个写法习惯，但非常流行，大家以后看到要认识。

接下来，就是量子计算的“动词”——**量子门**。你们在学数字电路或者编程时，一定知道逻辑门，比如“非门”（NOT gate）能把0变成1，1变成0；“与门”（AND gate）需要两个输入都是1，输出才是1。这些是经典计算的基本操作单元。量子计算也有类似的东西，叫做“量子门”。但它们做的事情不是简单的“翻转”，而是对我们的状态向量进行**旋转或翻转**。

还记得那个在单位圆上的向量指针吗？一个量子门操作，就相当于把这个指针从一个位置旋转到另一个位置。比如，作者提到的**阿达马门（Hadamard gate）**是一个非常基础且重要的量子门。它的神奇之处在于，可以把一个确定的状态变成一个完美的叠加态。如果你把处于 |0⟩ 状态（指针指向x轴）的量子比特送进阿达马门，它会把指针旋转到45度角的位置，让它变成50%的0和50%的1的叠加态。这就是创造“量子并行性”的第一步。一个完整的量子算法，就是一连串精心设计的量子门操作，像一套复杂的体操动作一样，精确地将初始状态向量旋转到我们想要的最终状态，从而让正确答案的概率达到最高。【原文】
它的作用是将水平的零方向上的单位向量映射到东北对角线方向，
并将垂直的一方向上的单位向量映射到东南对角线方向。
你通常会用它来将一个确定性状态，
即一个要么是零要么是一的状态，转变为一个 50-50 概率均等的状态，
反之亦然。
这只是一个例子，但还有许多其他的门构成了量子计算的基础模块，
在这种背景下，编写算法的艺术就是以某种方式将一堆不同的量子门组合在一起，
从而逐步地操Gesamtheit操纵、翻转和调整这个向量，直到它几乎完全指向某个特定的坐标方向，
而这个方向，想必就是能回答你所关心问题的那个方向。
回到最简单的量子比特的例子，
你只有两个坐标方向可以操作，
所以你只能回答简单的“是”或“否”问题。
尽管我无法画出超过三维的几何向量，
但原则上，一个有 k 个量子比特的系统将有 2 的 k 次方个
不同的坐标方向，每个方向对应一个比特串。
所以，如果你能设法让这个向量只指向其中一个方向，
你就可能回答一些更有趣、携带更多信息的问题。
也许其中一个方向代表了你试图分解的一个非常大的数的质因数，
或者代表了本视频开头谜题中的那个密钥值。
尽管量子计算机的潜在能力一直以来都有被过分夸大的传统，
但就其确实存在的更强潜力而言，
一个关键原因就是状态向量的大小是指数级增长的。
少至 100 个量子比特就已经意味着一个大到令人难以置信的状态向量。
但问题在于，你无法直接访问这个向量内部的值。
它对你来说实际上是不可见的。它能有用的唯一方法是，你有一种方法来操纵它，使得所有的概率，或者至少是大部分概率，都集中在单个分量上，并且这个分量对应于你关心的问题的答案。

【解读】
同学们好！我们来一起探索一下量子计算的奇妙世界。这段文字的核心，其实是在用一个我们非常熟悉的工具——向量，来描述量子世界的基本单元：量子比特（qubit）。

首先，忘掉那些深奥的物理学，我们来做个类比。想象一下你面前有一个二维坐标系，有 x 轴和 y 轴。一个普通的比特，就像一个开关，要么在 x 轴的正方向上（我们称之为状态 0），要么在 y 轴的正方向上（状态 1），泾渭分明。但量子比特不一样，它是一个**状态向量**，可以指向这个坐标系里的**任何方向**。当它指向 x 轴时，它就是 0；指向 y 轴时，它就是 1。但它也可以指向东北 45 度角的方向，这时它就处于一种“既是 0 又是 1”的叠加态。原文提到的“哈达玛门”就是一种量子操作，它就像一个神奇的旋转器，能把一个明确指向 x 轴的向量（确定状态 0），“啪”的一下旋转到东北方向，让它进入 0 和 1 概率各占 50% 的叠加状态。

现在，我们把维度升高。经典计算机里，8 个比特只能表示 256 个数字中的一个。但在量子世界，8 个量子比特所构成的状态向量，是生活在一个 **2 的 8 次方（256）维**的空间里！这个向量可以同时“指向”所有 256 个方向，每个方向都代表一个可能的经典结果。这就是量子计算力量的来源：**指数级增长的状态空间**。原文提到 100 个量子比特，那对应的状态向量就生活在一个 2 的 100 次方维度的空间里，这个数字比宇宙中所有原子的总数还要多得多！这意味着一台量子计算机原则上可以一次性处理海量的并行信息。

但这里有个关键的“陷阱”：你不能直接“偷看”这个向量到底指向哪里。在量子世界，任何“观测”行为都会迫使这个向量瞬间“坍缩”，随机地指向其中一个坐标轴。这就好比你有一个神奇的、包含所有可能答案的“魔法球”，但只要你看它一眼，它就立刻碎裂，只给你留下其中一块碎片。那么量子计算的意义何在呢？原文给出了答案：编写量子算法，就是设计一套精巧的“旋转”和“调整”操作（通过各种量子门），引导这个状态向量，在我们观测它之前，就让它以极高的概率指向我们想要的那个“答案”方向。这样一来，当我们最终“看”它的时候，它就极大概率坍缩到正确的答案上。这就是量子算法的艺术所在——在不直接看到答案的情况下，巧妙地操纵概率。

【原文】
Grover 算法为我们提供了一个很好的例子来实际看看这是如何运作的，现在是时候来了解它了。
让我为你提供一个非常高层次的预览，看看它是如何运作的。
我保证稍后会更详细地解释这一切，但现在先来鸟瞰一下。
它初始化这个状态向量，使得所有可能结果的概率均等分布。
其中一个结果将是你正在寻找的密钥，
而你将拥有的工具（我保证稍后会解释其动机）是在该坐标处翻转状态向量的符号。
这不会立即影响概率，
但当你将此操作与某个其他操作交错进行，在这两者之间来回切换时，

发生的情况是，概率质量会慢慢开始集中到那个
密钥值上，到了某个点，几乎所有的概率质量都会在那里，
所以当你从计算机中读取结果时，你几乎肯定会看到你正在寻找的密钥。
好的，以上是高层次的概述，但让我们更详细地展开它。
首先要解决的是这个想法：翻转与密钥相关联的分量的符号。
这可能感觉有点奇怪。
我们为什么会假设这个操作对我们是可用的呢?
回过头来，请记住，Grover 算法旨在应用于任何可以
快速验证解决方案的问题，即使找到解决方案本身很困难。
这里的例子包括解决数独、为地图找到一种有效的着色方案
（其中任意两个相邻区域的颜色都不同），
或者密码学中无数的任务，在这些任务中，安全性通常依赖于
某个值难以找到，尽管出于实用性考虑，它必须
易于验证。
我们在本视频开始时，用一个通用的替身来代表所有这些问题，
在这个替身问题中，你想象一个函数，它接收从
0 到 n-1 的任何数字作为输入，并且只对其中一个输入返回 true。

【解读】
好了，现在我们来聊聊一个具体的、非常酷的量子算法——Grover 算法。你可以把它想象成量子世界里的“超级搜索引擎”。它的目标是在一个巨大的、未排序的数据库中寻找一个特定的项目。比如，在一部有百万个电话号码的电话簿里，只知道一个人的名字，要找到他的号码。经典计算机怎么办？一个一个地查，平均需要查一半，也就是五十万次。而 Grover 算法能实现平方根级别的加速，大大减少搜索次数。

那么，这个算法的“魔术”是什么呢？原文用一个高层次的视角为我们揭示了它的核心步骤，我用一个比喻来帮你理解。想象一个巨大的体育场，里面坐满了观众（代表所有可能的结果），你要找的“密钥”就是其中唯一一个戴着特殊帽子的人。

第一步：**初始化**。算法开始时，我们让体育场里响起同样分贝的背景噪音，每个观众发出的声音都一样大。在量子世界里，这就是让状态向量处于一个“均匀叠加态”，每个可能结果被测量到的概率都完全相等。此时，你想找到那个特殊观众，就像在均匀的噪音里分辨出一个特定的声音，几乎不可能。

第二步：**标记答案（翻转符号）**。这是最关键的一步。我们有一个神奇的“探测器”，它能瞬间识别出那个戴着特殊帽子的人。但它不做别的，只是悄悄地给这个人发出的声音加上一个“反相”效果。想象一下，如果原来的声波是波峰，现在就变成了波谷。对于我们人耳来说，音量

（概率）没有变化，但声波的“相位”却被翻转了。这个操作在物理上并不会让那个人的声音变大。

第三.步：**放大差异**。接下来，我们执行另一个全局操作。这个操作可以想象成体育场的音响系统做了一次特殊的“回声处理”。它会把所有观众的声音，以“平均音量”为基准进行一次反转。结果会发生什么奇妙的化学反应呢？那些音量和平均值差不多的普通观众，他们的声音变化不大。但那个被我们做了“反相”标记的特殊观众，他的声音现在和“平均值”的差距变得非常大，经过这次反转，他的声音会被极大地增强！

通过不断重复“标记”和“放大”这两个步骤，那个特殊观众的声音（也就是我们找到正确答案的概率）会一轮轮地被放大，而其他人的声音则会被压制。最终，他的声音响彻整个体育场，我们一“听”（测量），就能准确地找到他！

原文还解释了为什么我们可以做到第二步的“标记”。因为 Grover 算法专门解决那些“答案难找，但验证容易”的问题，比如数独。你很难填出答案，但给我一个填好的数独，我一眼就能验证它对不对。这个“验证”过程，在量子计算机里就可以被转化成那个神奇的“符号翻转”操作。

【原文】
原则上，我们会认为这样的函数是由
一堆经典逻辑门构建的。
这些逻辑门作用于输入的某种二进制表示，
最终的输出是 0 或 1。
现在，关键点来了。
Grover 知道，对于任何像这样的逻辑门组合，
你都可以将其转换成一个量子门系统，使得如果在经典情况下，
函数接收某个二进制输入并返回 1（代表 true），
那么在量子情况下，所有这些门的效果就是翻转那个状态的符号，即与相同比特串相关联的状态。
类似地，如果在经典情况下，函数将某个二进制输入映射到 0（代表 false），
那么在这个量子转换中，对……的影响对应的状态将保持不变。
更一般地说，因为所有这些量子操作都是线性的，
如果状态是多个纯坐标方向的组合，
那么其效果就是简单地翻转与任何
触发该经典函数的比特串相关联的分量的符号。
这意味着，如果你有任何 NP 问题，任何可以快速验证
解的问题，你就能在量子计算机上创建一个操作，该操作可以翻转
与该问题解对应的状态向量位置的符号。

起初这可能感觉没什么用。
毕竟，翻转符号不会影响概率。
但格罗弗（Grover）意识到，这可以与
另一个步骤结合使用，该步骤会慢慢放大那个关键值的概率。
实际上，有一种非常好的方法来可视化他的算法。
为了进行设置，让我们想象我们的状态向量只有三个维度。
显然，原则上它会大得多，但这能让我画出第一张图。
这里的三个方向将对应于值 0、
1 和 2，而整个问题陈述是，这些值中的一个
是我们正在寻找的密钥。

【解读】
这一段深入探讨了 Grover 算法中最神奇一步的实现原理，也就是我们之前比喻的“给正确答案做标记”。它回答了一个核心问题：我们凭什么能拥有这样一个“黑科技”工具，它能恰好翻转正确答案所对应状态的符号呢？

答案出人意料地与我们熟悉的经典计算机逻辑有关。大家知道，经典计算机里的所有运算，追根溯源都是由“与门”、“或门”、“非门”这些基本的**逻辑门**组合而成的。比如，我们写一个程序来验证一个数独的解是否正确，这个程序最终会被编译成一长串作用于二进制输入的逻辑门操作。输入一个解（一串 0 和 1），经过这套逻辑电路，最终输出一个比特：1 代表“正确”，0 代表“错误”。

现在，关键的转换来了。量子计算领域有一个非常强大的结论：**任何经典的逻辑电路，都可以被转化为一个等效的、可逆的量子门电路**。Grover 算法巧妙地利用了这一点。他设计了一个通用的方法，能把那个用于“验证答案”的经典电路，变成一个我们称之为“神谕”（Oracle）的量子模块。这个“神谕”的作用非常精准：

- 当你把一个代表“错误答案”的量子状态输入给它时，它什么也不做，原封不动地输出。
- 当你把一个代表“正确答案”（也就是那个能让经典验证电路输出 1 的答案）的量子状态输入给它时，它就会给这个状态的“振幅”乘以 -1，也就是**翻转它的符号**。

更厉害的是，由于量子世界的**线性叠加原理**，如果你的输入是所有可能答案的叠加态，这个“神谕”会同时作用在每一个状态上。它会像一个精准的筛子，把所有“错误答案”放过，但唯独给那个隐藏在其中的“正确答案”贴上一个-1 的标签。

所以，这个看似神奇的“标记”能力，并非凭空而来。它建立在一个坚实的基础上：只要你的问题是一个可以被经典计算机快速验证的问题（这类问题在计算机科学中被称为 NP 问题），你就能构建出对应的量子“神谕”。

一开始，你可能会觉得这个“翻转符号”的操作没什么大不了。一个向量分量的符号从正变到负，它的平方（代表概率）并没有改变。就像我们之前比喻的，声音的相位变了，但音量没变。然而，Grove r的过人之处就在于，他意识到这个看似微小的相位变化，可以和另一个全局操作（我们后面会讲的“扩散”操作）相结合，产生滚雪球效应，最终将微弱的信号放大成主导性的信号。为了更直观地理解这个过程，原文提议，让我们从一个极简的三维向量空间开始，来画图看看这个“放大”过程到底是如何发生的。

【原文】
许多不同的量子算法都会首先将状态向量置于
一种均匀平衡的状态，其中所有分量都具有相同的值。
我想给那个均匀平衡的向量起个名字。
我们称之为 B。
我希望你不要太反对我只是简单地宣称这是可能的，
而没有详细说明实现它的底层量子门。
在这种情况下，它基本上看起来像一大堆哈达玛门（Hadamard gates），
但你只需要知道这个均匀平衡的方向是完全可以实现的。
所以从这里开始，目标是以某种方式迫使这个
向量转而指向那个密钥的方向。
在这里，对可视化非常有帮助的是，
在整个格罗弗算法中，该向量始终只在
由这两个向量张成的二维平面内移动。
所以我要做的是将所有东西都画在那个二维切片上。
即使当
完整维度大到我无法直接画出来时，这也将为我们提供一个忠实的表示。
我在这里绘制那个切片时将使用的惯例是，将密钥
方向，无论它是什么，沿着这个 y 轴放置，
然后 x 轴将代表所有
其他状态，即非密钥状态的均匀平衡。
所以在我们这个非常小的三维例子中，
如果那个密钥是 z 方向上的状态 2，
那就意味着这个垂直方向是 0 和 1 的均匀平衡，
它位于与 z 轴垂直的 xy 平面上。
请注意，完全均匀平衡的状态 b，在那个密钥
方向上有一些分量，因为根据其定义，它内部包含了一点那个密钥。
与其从三维空间中绘制这个切片，如果从更高维度来看，它看起来是这样的。
它几乎完全相同，但主要区别在于，那个均匀叠加态向量越来越接近于与密钥方向垂直。

但关键是，这个角度永远不会恰好是90度，
因为那个平衡态中总包含着一点点密钥值。

【解读】
好了，准备好进入 Grover 算法最直观、最优美的部分——几何可视化。我们之前说过，一个拥有 N 个可能答案的量子系统，其状态向量是生活在一个 N 维的复杂空间里的。直接想象这个空间对我们来说太难了。但 Grover 算法有一个惊人的特性，整个复杂的操作过程，自始至终都发生在一个**二维平面**上！这就像孙悟空有七十二变，但他的所有打斗都恰好发生在一个固定的舞台上。这一下，问题就从一个高维的怪物，简化成了我们高中就非常熟悉的平面几何问题。

这个关键的二维平面是由哪两个向量张成的呢?

1. **第一个基向量：正确答案的方向**。我们把所有可能答案想象成空间中互相垂直的坐标轴。我们要找的那个“密钥”或“正确答案”，就定义了其中一个坐标轴。为了方便，我们把它想象成这个二维平面的 **y 轴**。这个向量我们称之为 `|w>`（w for winner）。
2. **第二个基向量：所有错误答案的“平均”方向**。我们将所有除了正确答案以外的、其余 N-1 个“错误答案”的方向，做一个均匀的叠加。这个合成的向量就代表了“所有错误答案”的整体趋势。我们把它作为这个二维平面的 **x 轴**。这个向量我们称之为 `|s'>`。

现在，整个算法就可以在这个由 `|w>` 和 `|s'>` 定义的平面上展开了。

我们的**起始状态**是什么？还记得吗，是所有 N 个答案的均匀叠加态，原文称之为向量 B。这个向量 B 显然既包含了正确答案的成分，也包含了所有错误答案的成分。所以，在我们的二维平面上，这个起始向量 B 既不在 x 轴上，也不在 y 轴上，而是位于它们之间的一个**初始角度**上。

这里有一个非常重要的细节：当总答案数量 N 变得非常非常大时，这个起始向量 B 会**极其靠近 x 轴**（代表“错误答案”的平均方向）。这很好理解，因为在 N 个选项里，只有一个是正确的，剩下 N-1 个都是错误的。所以，这个“平均”向量 B，其绝大部分成分当然都来自于错误答案。但是，它和 x 轴之间永远不会完全重合，总会有一个微小的夹角，因为它里面终究“混入”了那么一点点正确答案的成分。

Grover 算法的全部精髓，就在于如何通过一系列的几何变换（其实是两次反射操作），将这个几乎躺在 x 轴上的起始向量，一步一步地“旋转”到 y 轴（正确答案）的方向上去。这个微小但不为零的初始角度，就是整个算法能够成功的起点。

【原文】
实际上，计算这个角度对于理解Grover算法的运行时间至关重要。
这是你为此真正需要做的主要数学计算。

你可以通过计算平衡态和密钥方向的点积来求得这个角度。
那个平衡态向量的所有分量都将是
1除以根号n，因为别忘了，所有分量的平方和必须等于1。
由于密钥向量除了一个分量是1之外，其他分量几乎都是0，
这意味着这两者之间的点积是1除以根 ѕιϥn。
如果你对点积有所了解，你就会知道，
这个值也是这两个向量之间夹角的余弦值。
为了方便我们之后使用，我将对这个事实稍作转换。
这等同于说，这里的这个余角的正弦值，
我称之为theta，是1除以根号n。
对于非常小的角，theta的正弦值和theta本身近似相等，
所以如果n非常大，我们可以有把握地说，这个角度大约是1
除以根号n，只要它是用弧度来度量的。
theta的这个值最终就是运行时间里那个平方根
的来源。
所以我们记住它。

【解读】
接下来，我们来做一点点数学，这部分非常关键，因为它直接揭示了 Grover 算法为什么能实现“平方根加速”的秘密。我们要计算的，就是上一部分提到的那个初始向量 B 与“错误答案平均方向”之间的微小夹角，我们称之为 `θ` 。

还记得高中数学里怎么求两个向量的夹角吗？对，用**点积**！公式是 `A · B = |A| * |B| * cos(α)` ，其中 `α` 是夹角。在量子力学中，所有的状态向量都是单位向量，也就是它们的模（长度） `|A|` 和 `|B|` 都等于 1。所以，点积就直接等于它们夹角的余弦值 `cos(α)` 。

我们来计算一下初始均匀叠加态向量 B 和正确答案向量 `|w>` 的点积。

- **正确答案向量 `|w>`** ：它代表一个确定的答案，比如在 N 个选项中，它是第 k 个。那么这个向量在一个 N 维坐标系里，只有第 k 个分量是 1，其他所有 N-1 个分量都是 0。它是一个标准的单位基向量。
- **初始向量 B**：它代表所有 N 个答案的均匀叠加。为了保证向量总长度为 1，根据勾股定理（所有分量的平方和等于 1），它的每一个分量都必须是 `1/√N` 。所以，B = (`1/√N` , `1/√N` , ..., `1/√N`)。

现在计算它们的点积 `B · |w>` 。这等于两个向量对应分量乘积之和。因为 `|w>` 只有第 k 个分量是 1，其余都是 0，所以点积的结果就是 `(1/√N) * 1` ，也就是 `1/√N` 。

这个点积 `1/√N` 等于 B 和 `|w>` （y 轴）之间夹角的余弦值。但是，我们更关心的是 B 和 `|s'>` （x 轴）之间的夹角 `θ` 。这两个角是互余的，所以 `cos(B与w的夹角) = sin(θ)` 。因此，我们得到了一个至关重要的结论：

`sin(θ) = 1/√N`

当 N 非常大时（比如在海量数据中搜索）， `1/√N` 是一个非常小的数，这意味着 `θ` 是一个非常小的角度。在高数中我们学过，当角度 `θ` （以弧度为单位）很小时， `sin(θ) ≈ θ` 。所以，我们可以近似地认为：

`θ ≈ 1/√N` (弧度)

这个公式就是 Grover 算法的核心！它告诉我们，算法的起始点距离目标（完全错误）只有 `1/√N` 这么一个微小的角度可以利用。而 Grover 算法的每一步操作，都能将这个状态向量向着正确答案的 y 轴旋转大约 `2θ` 的角度。那么，要从接近 x 轴旋转到接近 y 轴（大约 90 度，即 `π/2` 弧度），需要多少步呢？总角度除以每一步转动的角度，即 `(π/2) / (2θ) ≈ (π/4) / (1/√N) = (π/4) * √N` 。

看！运行的步数，或者说时间复杂度，是和 `√N` 成正比的，而不是经典算法的 `N` 。这就是“平方根加速”的数学来源，它完全来自于这个初始角度的几何计算。好的，同学们，今天我们来挑战一个非常前沿且激动人心的话题——量子计算。别担心，我们不会陷入复杂的物理公式，而是会用一种非常直观、甚至可以说是“眼见为实”的方式，来理解一个著名的量子算法：Grover搜索算法。它就像是为海量数据设计的超级搜索引擎。现在，请跟随我的思路，我们将一步步揭开它的神秘面纱。

【原文】
我把它放在这里的角落里。
好了，花了一些时间构建好实际的图像后，
让我来向你展示这里的步骤。
它出奇地简单。
还记得我们之前看的那个关键操作吗?
就是那个，你为你试图解决的任何NP问题取其验证函数，
并将其转化为一些量子门，
其效果是翻转与密钥值相关联的符号?
那么，在我们的图示中，这看起来是什么样的呢?
在这个图示中，如果你翻转与密钥值相关联的分量的符号，
而所有其他分量保持不变，其效果看起来就像是绕x轴翻转。
你需要知道的最后一个要素是，也可以
将这个态向量绕着这个均匀叠加方向进行翻转。

我意识到，我一直声称某些
操作是可行的，可能会让你们有点不满意，但有一点需要知道的是，总的来说，
如果你能清楚地描述并访问其中一个态向量，
那么绕它进行反射也是完全可能的。
有一些量子门可以让你绕着这个平衡方向翻转，
但请相信我，纠结于这些量子门具体是什么样的细节，并不会为整个算法
增加多少清晰度或直观理解。这个见解其实就源于几何学。
注意，如果你先围绕 x 轴进行翻转，
然后再围绕这个非对角线方向进行翻转，
这个状态现在会更偏向垂直方向一点。
如果你现在从内存中读取，你看到密钥值的机会就会比看到所有其他值的机会稍大一些。
在这里，如果我展示出实际的坐标，也许会更有帮助，
在这个 n 等于 100 的例子中，同时展示基于这些坐标平方的概率条。
注意，每当我围绕 x 轴进行翻转，
然后再围绕这个非对角线轴进行翻转时，与那个密钥相关联的分量就会变大一点。
所以，Grover 算法非常简单。
你要做的就是一遍又一遍地重复这个过程，直到向量尽可能地指向密钥方向。

【解读】
同学们，大家好！请想象一下，我们正在玩一个“大海捞针”的游戏。这个“大海”里有N个一模一样的储物柜，只有一个柜子里藏着宝藏。在经典世界里，你最好的方法就是一个一个地打开看，平均需要打开 N/2 次才能找到。但如果 N 是一个天文数字，比如宇宙中的原子数量，这就会变得不可能。Grover算法就是量子世界给出的一个巧妙解法。

首先，让我们把这个问题几何化。想象一个巨大的空间，它有N个维度，每个维度对应一个储物柜。我们可以用一个向量，或者说一个箭头，来表示我们当前的状态。一开始，我们对宝藏在哪一无所知，所以我们让这个“状态向量”公平地指向所有N个方向，这就是“均匀叠加态”。你可以把它想象成一个指向所有坐标轴正方向“正中央”的箭头。

现在，Grove算法的核心操作登场了，它只有两步，就像一个优雅的华尔兹舞步。

第一步，我们称之为“神谕”（Oracle）操作。我们有一个神奇的工具，它能“识别”出藏有宝藏的那个柜子对应的维度。当我们的状态向量经过它时，它会做一件很特别的事：将这个向量在“宝藏”维度上的分量符号翻转（比如从正变为负），而其他所有维度的分量保持不变。原文中将其比喻为“绕x轴翻转”，这是一个非常棒的二维简化。想象你的宝藏在y轴上，所有其他“错误答案”都在x轴上。这个操作就相当于把整个状态向量沿着x轴做了一个镜像反射。

第二步，是“绕均匀叠加方向翻转”。这一步听起来有点抽象，原文也请我们“相信”这是可以做到的。我们可以这样理解：还记得我们一开始那个指向“正中央”的初始向量吗？第二步操作就

是以这个初始方向为对称轴，对我们刚刚被第一步操作改变了的向量再做一次镜像反射。

现在，神奇的事情发生了！原文提到，“如果你先围绕 x 轴进行翻转，然后再围绕这个非对角线方向进行翻转，这个状态现在会更偏向垂直方向一点。” 这里的“垂直方向”就是我们宝藏所在的方向！这两次连续的反射，其净效果，其实是一次小小的“旋转”。每一次我们重复这一对“翻转再翻转”的操作，我们的状态向量就会向着正确答案的方向旋转一小步。这就好比我们每次都在微调我们的探照灯，让它越来越对准宝藏。Grover算法的精髓就在于，它告诉我们只需要不断重复这个“两步舞”，我们的状态向量就会无限逼近那个藏着宝藏的维度。最终，当我们进行“测量”时，就极大概率能找到它！

【原文】
你需要做的最后一步推理是，计算出应该重复多少次。
几何学中有一个奇妙的事实：如果你像这样连续地围绕两条不同的线进行翻转，其总体效果等同于一次旋转，
更具体地说，是旋转这两条线之间夹角的两倍。
所以在我们的例子中，相继应用我们可用的这两个操作，
净效应就是将状态向量旋转两倍的 theta 角，
这里的 theta 还是我们之前计算出的那个小角度，
大约是 n 的平方根分之一。
最终目标是将我们的初始状态旋转不到 90 度，
或者说大约二分之 pi 弧度。
这意味着最佳重复次数看起来像是二分之 pi 除以两倍的 theta，
也就是四分之 pi 乘以 theta 的倒数，而且关键的是，
因为 theta 大约是 n 的平方根分之一，
这意味着总步数看起来像是四分之 pi 乘以 n 的平方根。
所以 Grover 算法所说的是，首先找到最接近这个值的整数，然后将你可用的两个翻转操作重复那么多次。
举一个具体的例子，假设 n 是 2 的 20 次方，
这意味着你在大约一百万个选项中搜索一个密钥。
你会在一台至少有 20 个量子比特的计算机上运行这个算法，
而算法会告诉你，首先计算四分之 pi 乘以这个数的平方根，结果大约是 804，
这意味着你现在要重复那两个你可用的操作，
即将密钥状态的符号翻转的操作，以及围绕平衡方向翻转的操作，重复 804 次。
现在请记住，这个状态向量对你来说是不可见的。
你无法做任何事来读出它所具有的值。
你必须通过推理来推断它必定在哪里，而在本例中，
通过所有的几何推理，你可以得出结论：在重复了特定次数之后，该向量应该几乎完全指向

那个密钥的方向。
所以，当你从计算机中读取结果时，你几乎肯定会看到那个密钥值。需要澄清的是，你并不一定能看到它。
在读取之后，有很小的可能性，
你会看到一个不是密钥的东西。
所以为了确保万无一-失，你总是可以快速验证答案，毕竟，
整个情景的前提就是你有一种快速验证答案的方法。
你可以在经典计算机上做到这一点。
最坏的情况下，如果你运气不好，采样到了一个不同的数字，
你可以把整个过程再运行一遍，这样一来，
你需要运行超过几次的可能性就变得微乎其微了。

【解读】
好的，我们已经知道了Grove算法的核心“舞步”——两次反射等于一次旋转。现在，一个至关重要的问题摆在我们面前：这个舞要跳多少次？跳少了，我们的状态向量还没转到终点；跳多了，可能就转过头了。这就是这一部分要解决的问题。

原文揭示了一个美妙的几何学原理：两次连续的反射，等于一次旋转，旋转的角度是两条反射轴之间夹角的两倍。在我们的算法里，初始状态（均匀叠加态）和宝藏状态（密钥）之间的夹角非常小，大约是 1/√N 弧度（我们称之为θ）。因此，我们每重复一次“两步舞”，状态向量就会向着正确答案旋转 2θ 的角度。

我们的目标是什么？是从几乎与正确答案“垂直”（相差约90度，即π/2弧度）的初始状态，旋转到与正确答案完全重合。那么，总共需要旋转的角度就是 π/2。用我们需要的总角度除以每一步能旋转的角度，就能得到需要重复的次数：(π/2) / (2θ) = (π/4) * (1/θ)。因为 θ 大约是 1/√N，所以总步数大约是 (π/4) * √N。

这就是Grover算法最惊人的地方！请注意这个 √N。在经典世界里，从N个选项里找一个，平均需要 N/2 次。而在量子世界里，我们只需要大约 √N 次。这个从N到√N的飞跃，就是所谓的“量子加速”。

我们来看原文中的例子：N = 2^20，这大约是一百万。经典搜索平均需要50万次。而量子算法需要多少次呢？(π/4) * √(2^20) = (π/4) * 2^10 ≈ 0.785 * 1024 ≈ 804 次！从50万次到804次，这是一个巨大的效率提升。

这里有一个非常关键的量子力学概念需要大家理解：在整个计算过程中，这个状态向量对我们来说是“不可见的”。我们不能像看一个钟表指针那样实时观察它旋转到了哪里。我们只能在算法结束时，进行一次“测量”（原文中叫“从计算机中读取结果”）。测量这个行为，会迫使这个处于叠加态的向量“坍缩”到一个确定的状态上。而它坍缩到哪个状态的概率，取决于它离那个

状态有多“近”。经过我们精确计算的804次旋转后，状态向量已经几乎完全指向了“宝藏”的方向，所以当我们进行测量时，有超过99%的概率会直接得到正确答案。

当然，量子世界是基于概率的。总有那么一丝可能，我们运气不好，测量到了一个错误的答案。但别担心，因为这类问题（NP问题）都有一个特点：验证答案非常容易。我们把测量出的结果用经典计算机一验证，就知道对不对。如果不对，怎么办？很简单，把整个量子算法再运行一遍。由于单次成功率极高，重复几次就能确保找到正确答案，总的计算成本依然远低于经典搜索。

【原文】
最后，我想坦白一个我一直在对你们说的谎，
并稍微反思一下这种加速来自何处，
然后重点介绍一个惊人的类比。
在此之前，现在或许是向 Patreon 上
支持本频道的社群成员们道谢的好时机。
制作像这样的可视化课程需要花费大量时间。
你可能知道，大多数 YouTuber 通过视频内赞助来将内容变现，
但多年来我一直选择拒绝这种方式。
我认为这能让视频质量更高，而它之所以没有成为一个代价高昂的决定，
唯一的原因是，有足够多认同这一点的观众通过 Patreon 直接支持本频道。
作为回报，我为支持者们提供提前观看新内容的权益，
这实际上对内容的开发非常有帮助，此外还有一些其他福利。
总而言之，如果你喜欢这些内容，
并考虑加入支持者行列，那将意义重大。
不过别有压力，本频道的一大价值就在于内容可以免费观看。
总之，让我们回到最后那三点。
这个谎言是一个因疏漏而产生的谎言。
我一直在展示的状态向量都带有正负实数值，
但更普遍地，这些分量可以是复数。
我希望在接下来关于物理的视频中解释为什么会是这样。
总的来说，当你处理波的时候，
你既关心振幅也关心相位，而复数
是同时编码振幅和相位的一种非常优雅的方式。
所以，如果你看状态向量中的一个分量，
更全面的理解方式是，它既有某个模长
也有某个相位。
模长是那个你求平方后得到概率的值，

而相位本质上是我们
在此讨论的正负值的一个更普遍的版本。
改变相位不会直接影响概率，
但它会影响状态，这反过来又会影响它被处理的方式以及它
与世界互动的方式。
不用说，引入一堆复数会给
一个本已复杂的话题增加很多潜在的困惑。
这就是我之前避开它的原因。
但就 Grover 算法而言，我们可以安全地忽略复数值，
因为非常幸运的是，在该算法的执行过程中，
你只会看到正值和负值。不过，我希望你知道，这种复数值的可用性在其他量子算法中扮演着至关重要的角色，比如用于分解质因数的Shor算法。
其次，即使你理解了我所描述的整个算法的全部内容，也很难总结出加速究竟来自哪里。
你一开始就对这个均匀叠加态应用某种操作，这个事实让人很想说加速来自于对所有可能的输入并行执行该操作。
但正如我开头提到的，这种总结，至少对我来说，感觉真的不太对劲，而且肯定会导致误解。
正如你现在所知，单靠这一步本身，并不能揭示出关键值。
至于把这第一步描述为对多个输入并行应用一个函数是否恰当，或者说，认为这个叠加态本身就是一个全新的事物，而我们所拥有的函数始终是每次只作用于单个输入，从不同时作用于多个，这样感觉更好——我将把这个问题留给你自己来解读。
只不过现在这些输入是一种新的东西。
在我看来，对于这个算法，如果你想用一个词来总结加速的来源，我认为更好的选择是“毕达哥拉斯”。
打一个不那么严谨的比方，如果你想从一个单位正方形的一个角走到对角，如果你只能沿着x和y方向移动，你必须走两个单位的距离。
但如果你可以沿对角线移动，你走根号二的距离就能到达。
更一般地，如果你在n维空间中，要从一个立方体的一个角走到对角，如果只能沿着棱移动，你将需要走n个单位，但如果你可以走对角线，你走根号n的距离就能到达。
在量子力学的世界观中，不同的可观测态都代表了某个状态空间中的相互垂直的方向。
所以，从这个框架来看，经典的确定性世界就像一个你只能访问这些纯粹坐标方向的世界。

【解读】
在课程的最后，作者坦诚了一个为了教学便利而设下的“善意的谎言”，并深入探讨了量子加速的真正来源，最后给出了一个绝妙的比喻。让我们逐一来看。

首先，那个“谎言”是关于向量分量的。到目前为止，我们都假设状态向量的分量是正数或负数。这在理解Grover算法时是完全够用的，也更容易想象。但实际上，一个真正的量子态向量，它的分量是“复数”。同学们在数学课上应该学过复数，它既有大小（模），也有方向（相位）。我们之前说的“正”和“负”，其实只是相位为0度和180度的特殊情况。你可以想象每个分量都是一个钟表上的指针，既有长度，也能指向任意角度。这个“相位”在很多更复杂的量子算法（比如Shor算法）中扮演着至关重要的角色，它是量子计算强大能力的另一个源泉。不过，幸运的是，Grover算法的整个过程恰好只涉及0度和180度的相位，所以我们用正负实数来理解是没问题的。

其次，量子加速的来源到底是什么？一个非常流行但容易误导的说法是“量子并行计算”，即量子计算机一次性计算了所有可能的结果。但作者明确指出，这并不准确。仅仅把所有可能性叠加在一起，并不能直接告诉你答案。Grover算法的真正威力，来源于一种叫做“振幅放大”的机制，而我们之前学习的几何图像——两次反射等于一次旋转——正是对这种机制最直观的描述。它不是简单地并行计算，而是一种巧妙的“干涉”过程：通过精心设计的操作，让指向正确答案的“路径”被不断加强（建设性干涉），而指向错误答案的“路径”被相互抵消（破坏性干涉）。我们的几何旋转，就是这个干涉过程的可视化结果。

最后，作者给出了一个我认为是理解量子加速最深刻、最精彩的类比：“毕达哥拉斯”。想象一下，你要从一个正方形的左下角走到右上角。如果是在一个城市街区，你只能沿着街道走（先向右再向上），总路程是 1+1=2。但如果你可以“走对角线”，直接飞过去，根据勾股定理（毕达哥Gougu拉斯），路程只有√2。现在，把这个思想扩展到N维空间。要从一个N维立方体的一个角走到最远的对角，如果只能沿着棱走（像经典计算机一样，一次处理一个维度/可能性），你需要走N步。但如果你能走N维空间中的那条“大对角线”，路程就只有√N！

这个类比告诉我们：量子计算的优势，源于它能够在一个更高维、更自由的“计算空间”里运行。经典计算机被限制在只能沿着坐标轴移动，而量子计算机则可以利用叠加和纠缠，开辟出我们之前无法想象的“对角线”捷径。Grover算法的√N步，正是走了这样一条经典世界里不存在的“计算捷径”。这或许就是量子世界赋予我们的、看待和解决问题的全新视角。好的，作为一位专业的学术导师，我将为您详细解读这份关于量子计算的Markdown文档。我会将复杂的概念用生动有趣的方式，结合您在高中阶段所学的知识，进行深入浅出的讲解。

【原文】
如果你仔细想想，任何时候进行计算，你的计算机都在以某种方式穿行于一系列它可用的不同状态，而算法的运行时间就是要理解你必须在所有可能状态的空间中走多少步。
因此，从这种量子世界观来看，经典状态看起来就像纯粹的坐标方向，而量子计算的关键区别在于，你现在还可以使用大量额外的对角线方向。
现在要澄清一下，这个类比并非完全字面意义上的，你应该持保留态度地看待它。
这并不是说运行时间就一定像是穿越这个特定状态空间的距离。

但事实确实是，如果你在整个Grover算法中跟踪状态向量，它所做的就是从一个初始条件缓慢地沿着一个四分之一圆弧走向目标条件，描绘出一条完全……如果你被限制只能沿着纯粹的坐标方向移动，这是无法实现的。
其效果就是提供了这条大小为平方根的捷径。
在结束之前，作为最后一点，
老观众们会知道，我之所以讲这个主题，原因之一是之前承诺过的一个类比，
这个类比是关于本算法和我们上期视频讲过的内容，
也就是两个可以计算pi的碰撞方块。
在那个视频里，我们同样研究了一个在特定二维状态空间中绕着一个圆弹跳的点，
事实上，它所经历的一系列弹跳与我们刚才在Grover算法中看到的本质上是相同的。
这里的故事是，当我的一位物理学家朋友亚当·布朗 (Adam Brown)
看到那个视频的初版时，他最近正好在研读Grover算法，
并立刻意识到这两个过程是完全相同的。
我曾为这个视频制定了一个十分荒唐的计划，
打算在那个类比的背景下解释量子计算和Grover算法，
但结果证明这真是个糟糕透顶的主意。
整件事只有在你已经理解了Grover算法的情况下才真正有意义。
所以，既然你们已经独立地看过了这两个主题，
我要做的是留下一个关于这个类比是怎样的粗略提纲。
把这看作一道开放式的家庭作业谜题，
任务是让你自己去建立其中的联系。
作为某种形式的答案，我会附上亚当·布朗写的一篇非常有趣的论文的链接，
它精确地概述了那个类比具体是什么样的。

【解读】
同学们，大家好！我们来一起解开这段文字的奥秘。这段话的核心，是用一个非常巧妙的比喻，来解释量子计算为什么可能比我们日常使用的经典计算机更快。

首先，让我们想象一下计算过程。你可以把计算机的每一个瞬间状态，看作是地图上的一个点。比如，内存里存着数字5，这就是一个状态；下一秒，它变成了6，这就是另一个状态。一个“算法”，就像是在这张巨大的“状态地图”上规划一条从起点（问题）到终点（答案）的路线。算法的“运行时间”，就相当于你走完这条路需要多少步。

现在，经典计算机的“地图”有什么特点呢？作者说，它就像一个城市街区，你只能沿着东西向或南北向的街道走，也就是文中所说的“纯粹的坐标方向”。如果你想从西南角走到东北角，你必须先向东走几个街区，再向北走几个街区，走的是一条折线。

而量子计算的“地图”则完全不同。它允许你“走对角线”！想象一下，你可以直接从西南角拉一条直线，穿过所有街区直达东北角。显然，这条对角线捷径比沿着街道走要短得多。这就是量子计算的魅力所在——它开辟了无数条经典计算机无法行走的“对角线路径”，从而可能找到更短的“解题路线”。文中提到的“Grover算法”（一种著名的量子搜索算法），它在状态空间里的运行轨迹，正是一条优美的“四分之一圆弧”，这是一条经典计算机无法走出的捷径。这条捷径带来的速度提升，正是“平方根”级别的。打个比方，如果经典计算机需要在大约100万个数据里搜索，平均要找50万次；而Grover算法只需要大约1000次（√1,000,000），效率天差地别！

更有趣的是，作者提到了一个看似毫不相关的例子：“两个碰撞的方块计算π”。这听起来很神奇，但其背后的数学原理，和Grover算法在状态空间中的演化，竟然是“本质上相同的”！这揭示了科学中一个深刻而美妙的道理：不同的物理现象背后，可能隐藏着相同的数学结构。就像抛物线既能描述你扔出的篮球的轨迹，也能描述二次函数的图像一样。作者在这里留下了一个悬念，鼓励我们像真正的学者一样，去主动探索和发现这两个看似风马牛不相及的领域之间隐藏的深刻联系。这不仅仅是在学习知识，更是在体验科学发现的乐趣。

【原文】
如果你想学习更多量子计算的基础知识，
几年前，我的两位非常聪明的朋友，安迪·马图什查克 (Andy Matuszczak) 和迈克尔·尼尔森 (Michael Nielsen)，
制作了一个非常好的学习资源，
提供了一种相当独特的方法来确保你能够长期记住这些知识。
要学习一些基础的量子力学，
来自Looking Glass Universe频道的Mithina Yoganathan
一直在制作一门关于这个主题的、对初学者非常友好的课程。
实际上，很多年前就是她教会我Grover算法是如何工作的，
说实话，这个视频的很多内容都要归功于与她的多次有益的谈话。
作为结尾的最后一点补充，我在制作这个视频期间的一次谈话
是和斯科特·阿伦森 (Scott Aaronson) 进行的，
他是一位备受尊敬的研究员，也是一位就此主题著书的、非常有趣的作者。
我录下了这次Zoom通话用来做笔记，其中有一个小片段
我想和你们分享。
你知道吗，我梦想写一部科幻小说，而且我已经想好了高潮场景，
主角们会，你知道的，
运行Grover算法来试图找到这个加密密钥，对吧?
整个世界的命运都将取决于他们能否找到它，好吗?
坏人已经包围了他们的基地，你知道，他们正在砸墙。

但Grover算法还在运行，而此时测量，它只有大约
30%的概率能给出答案。所以问题是，你是现在就测量，还是让它再运行一分钟?
对吧?
而且如果你现在测量，而你没有得到结果，那你就前功尽弃了。
然后你就必须从头开始。
所以，这是任何经典算法都不可能有的情形。
没错。

【解读】
同学们，在探索了Grov_er算法的奇妙之处后，这段文字为我们指明了继续前进的道路，并用一个生动的科幻故事，揭示了量子计算一个最核心、也最反直觉的特性。

首先，作者非常负责任地为我们这些“入门者”推荐了学习资源。这在学术探索中至关重要。他提到了迈克尔·尼尔森（Michael Nielsen），这位可是量子计算领域的泰斗级人物，他的著作是无数研究者的必读经典。这告诉我们，学习任何高深知识，都要找到权威、可靠的源头，站在巨人的肩膀上才能看得更远。同时，他也提到了教会他Grover算法的朋友，这体现了学术交流与合作的重要性。知识的火花，常常在思想的碰撞中产生。

接下来，就是这段文字最精彩的部分——来自量子计算专家斯科特·阿伦森（Scott Aaronson）的科幻故事。这个故事不仅仅是为了有趣，它是一个绝佳的思想实验，让我们能切身体会量子算法的“脾气”。

想象一下电影里的场景：英雄们必须破解密码才能拯救世界，时间紧迫。他们使用的是量子计算机。经典计算机破解密码，就像一个个试钥匙，要么成功，要么失败，进程是确定的。但量子计算机不同。Grover算法在运行时，它并非在“寻找”答案，而是在“放大”正确答案的“概率”。在故事的那个瞬间，测量得到正确密码的概率是30%。

这就是英雄们面临的“量子赌博”：

1. **现在测量**：你有30%的几率瞬间成为救世主。但同时，你有70%的几率失败。而量子计算最诡异的地方在于，一旦测量失败，整个计算过程就会“坍缩”（Collapse），之前所有的努力都会烟消云散，一切归零，你必须从头再来。可敌人已经在砸门了，你没有第二次机会。
2. **再等一会**：让算法再运行一段时间，正确答案的概率可能会继续上升，比如到80%、90%。但问题是，你可能等不到那个时候，敌人就已经冲进来了。

这个两难困境，是经典计算机永远不会遇到的。它深刻地揭示了量子计算的几个核心概念：

- **概率性**：量子算法的输出本质上是概率性的，而不是确定性的。

- **测量问题**：在量子世界中，“观察”或“测量”这个行为本身，会不可逆地改变系统状态。在你“看”它之前，它处于多种可能性的叠加态；你一“看”，它就随机地选择了一个状态“定格”下来。
- **叠加态的毁灭**：一次失败的测量会摧毁宝贵的量子叠加态，让计算前功尽弃。

所以，这个故事告诉我们，量子计算不仅仅是“更快”的计算，它是一种全新的、基于概率和测量的计算范式，充满了不确定性和戏剧性。这既是它的力量所在，也是驾驭它的挑战所在。