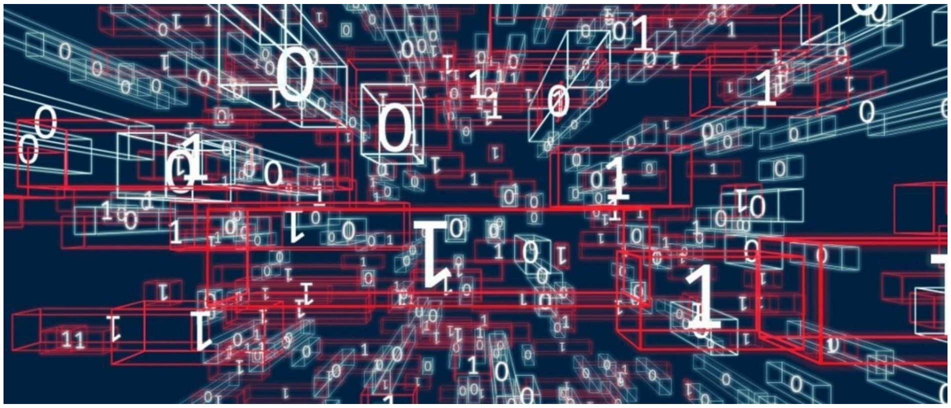
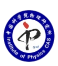
一文教你 “量子编程”入门式 - 知乎

笔记本:	量子计算机
创建时间:	2020/8/26 17:15
URL:	https://zhuanlan.zhihu.com/p/94015026



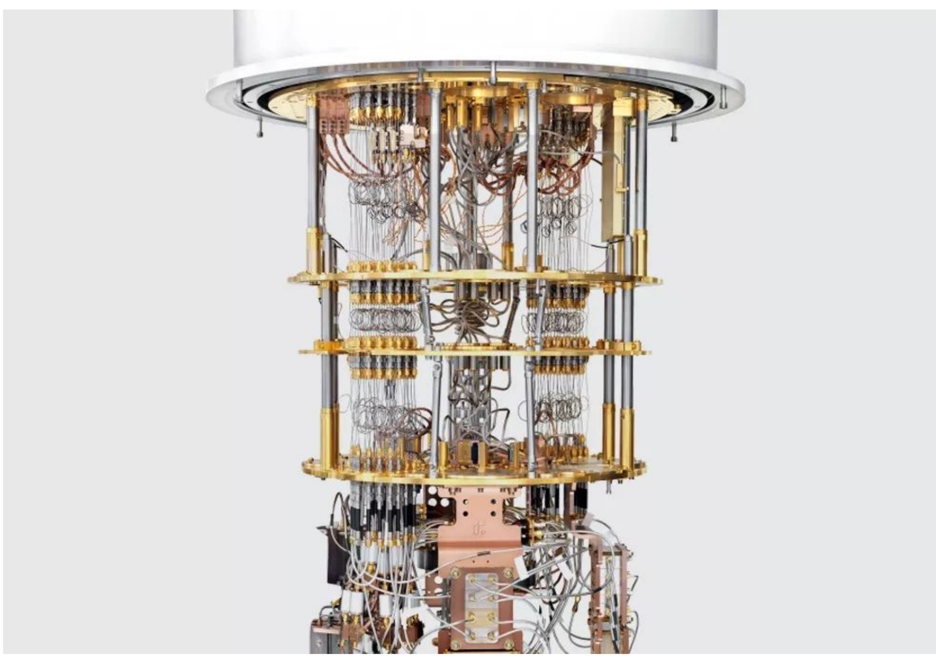
一文教你 “量子编程”入门式



中科院物理所
物理学话题下的优秀回答者

> 作者：Quentin Truong
> 翻译：Nuor
> 审校：YQH

这是一段从**量子比特**到真实**量子程序**的**量子编程演练。**



量子计算机

量子计算机被发现之后，量子编程也在不断发展。本文将带你**入门量子编程**，介绍**量子计算机与传统电脑的区别**，解释量子编程的基本概念，最后教你如何在一个当今免费的量子计算机上运行程序。

在开始之前，**请注意**，本文是为希望了解量子编程的完整技术细节的人们准备的。本文建立在**量子比特**（qubit），**量子门**（quantum gates）和**量子电路图**（quantum circuit diagrams）的数学基础之上。（本文不涉及解释量子算法。）

由于我们将涉及到有关量子编程的基础数学，因此你需要了解**向量、矩阵、线性组合和复数**的概念。

量子计算机

让我们首先了解一下**什么是量子计算机，其与传统的计算机有什么区别?**

量子计算机是使用量子力学进行运算的机器。

那么，这与其他计算机有何不同？我们知道，计算机最基本的形式是用来执行运算，有许多类型的计算机，在计算机时代的早期，我们实际上是有过**机械计算机**的——查尔斯·巴贝奇在1837年设计了一款机械计算机来执行基本的计算过程。现今，我们的计算机基于**数字电子设备**，利用**位**和**逻辑门**来进行运算。与之不同的是，量子计算机使用**量子力学**来进行运算，利用**量子比特**和**量子门**，而不是**位**和**逻辑门**。

那么，**什么是量子比特和量子门呢**？物理上来说，它们可以用许多方式实现——Google、IBM、微软、Rigetti等公司都有自己的方案。我们现阶段无需关心量子比特和量子门的物理性质，因为初次学习量子程序并不需要了解这些。

量子编程

在开始编程之前，强烈建议你**摒弃**大脑中有关编程的一切固有概念，**不要**想着声明设置变量，循环语句，定义函数等，任何先入之见都没有用。量子编程不是简单地将现在的程序运行更快的一种方法，其与现有的程序在根本上就是完全不同的。（按语：“量子编程”用是类似HDL的硬件描述语言，而不是如C之类的面向过程式语言。两者无法类比。）

了解量子比特

一个量子比特是具有单位长度的两份复数的矢量。为什么使用量子比特？量子比特的含义是什么？我们不妨把它和传统的位（即比特）作比较。

对于初学者来言，“一个位”是一个非0即1的数。而“一个量子比特”是一个非0即1的概率分布：若有两个相同的量子比特，分别测量，可以测到不同的值。仔细想一想，就可以发现，基于量子比特的量子计算，本质上也是概率性的！

第二个重要的区别是，一个传统的位可以被无限次读取，而量子比特一旦被读取过，就会失去量子性，坍缩成一个传统的位。想想薛定谔的猫，“测量”就是打开盒子，看看猫有没有死。一旦发现它死了，即使合上盖子再打开，猫也不会再活过来。用量子力学的话说，“波函数发生了坍缩”。

我们假设一个量子比特可以测到0的概率为|α|2，测到1的概率为|β|2。由于测到的值非0即1，所以：

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \text{ such that } |\alpha|^2 + |\beta|^2 = \alpha\cdot\overline{\alpha} + \beta\cdot\overline{\beta} = 1$$

其中，包含α和β的列向量是一个量子比特，α和β上方的横线表示复共轭。之所以把概率写成α和β模长的平方，是因为α和β本质上是两个波函数的幅值，而任何测量手段都是在取各自的模平方。

总结：**量子比特是两个复数的α和β的单位向量。量子比特被测量概率为0的是|α|2，被测量为1。α和β在被测量之前，是不得而知的，测过之后则坍缩。**

量子比特符号

我们通常使用**狄拉克标记**（也称作bra-ket标记）表示量子比特。bra代表行向量，用⟨ |表示，ket代表列向量，用| ⟩表示。例如，我们可以按照以下的方式在狄拉克表示中写入测到的“0”状态和“1”状态。（注意不要将bra/ket的表示和向量内的内容混淆。）

$$\langle 0| := \begin{pmatrix} 1 & 0 \end{pmatrix} \qquad |0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\langle 1| := \begin{pmatrix} 0 & 1 \end{pmatrix} \qquad |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

量子比特有**纯态**和**混合态**的区别。如果一个量子比特的态可以完全用**|0⟩和|1⟩**线性表示，这就是一个**纯态**：

$$|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

还有一些纯态的简单表示例子：

$$|+\rangle := \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$$

$$|-\rangle := \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle$$

$$|i\rangle := \frac{1}{\sqrt{2}} |0\rangle + \frac{i}{\sqrt{2}} |1\rangle$$

$$|-i\rangle := \frac{1}{\sqrt{2}} |0\rangle - \frac{i}{\sqrt{2}} |1\rangle$$

其他量子比特需要纯态的混合才能充分描述它们，被称为**混合态量子比特**。换句话说，混合态量子比特是通过纯态的概率分布来描述的。在本文稍后的部分，我们将看到一个混合态量子比特的示例。

多个量子比特

到目前为止，我们仅定义了单个量子比特的状态。**多个量子比特的组合状态是什么样的?**

多个量子比特的组合状态其实就是所有量子比特的张量积。

如果你不知道张量积是什么，请不要担心。我们将通过一个简单的示例来进行介绍（⊗是张量积运算的符号）。

$$|0\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}1\cdot\begin{pmatrix}0\\1\end{pmatrix}\\0\cdot\begin{pmatrix}0\\1\end{pmatrix}\end{pmatrix} = \begin{pmatrix}1\cdot 0\\1\cdot 1\\0\cdot 0\\0\cdot 1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

通常，我们可以通过以下两个步骤对任何两个矩阵进行张量积：

1. 将第一个矩阵中的每个元素乘以第二个矩阵；

2. 根据第一个矩阵中元素的原始位置，合并乘出来的矩阵。

这是一个如何处理二维矩阵的示例：

$$I \otimes H = \begin{pmatrix}1&0\\0&1\end{pmatrix} \otimes \begin{pmatrix}\frac{1}{\sqrt{2}}&\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}&-\frac{1}{\sqrt{2}}\end{pmatrix} = \begin{pmatrix}1\cdot\begin{pmatrix}\frac{1}{\sqrt{2}}&\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}&-\frac{1}{\sqrt{2}}\end{pmatrix} & 0\cdot\begin{pmatrix}\frac{1}{\sqrt{2}}&\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}&-\frac{1}{\sqrt{2}}\end{pmatrix}\\0\cdot\begin{pmatrix}\frac{1}{\sqrt{2}}&\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}&-\frac{1}{\sqrt{2}}\end{pmatrix} & 1\cdot\begin{pmatrix}\frac{1}{\sqrt{2}}&\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}&-\frac{1}{\sqrt{2}}\end{pmatrix}\end{pmatrix} =$$

$$\begin{pmatrix}1\cdot\frac{1}{\sqrt{2}} & 1\cdot\frac{1}{\sqrt{2}} & 0\cdot\frac{1}{\sqrt{2}} & 0\cdot\frac{1}{\sqrt{2}}\\1\cdot\frac{1}{\sqrt{2}} & 1\cdot-\frac{1}{\sqrt{2}} & 0\cdot\frac{1}{\sqrt{2}} & 0\cdot-\frac{1}{\sqrt{2}}\\0\cdot\frac{1}{\sqrt{2}} & 0\cdot\frac{1}{\sqrt{2}} & 1\cdot\frac{1}{\sqrt{2}} & 1\cdot\frac{1}{\sqrt{2}}\\0\cdot\frac{1}{\sqrt{2}} & 0\cdot-\frac{1}{\sqrt{2}} & 1\cdot\frac{1}{\sqrt{2}} & 1\cdot-\frac{1}{\sqrt{2}}\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0\\\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0\\0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{pmatrix}$$

我们还可以将狄拉克表示中的多个量子比特表示为$|0\rangle\otimes|1\rangle$。作为简写，我们可以省略⊗，仅写$|0\rangle|1\rangle$；更简略些，写成$|01\rangle$。

了解量子门

现在让我们了解一下**量子门。**

量子门是一个幺正矩阵。
为什么是幺正矩阵?

首先，量子门将由物理设备实现，因此它们必须遵守量子物理定律。物理学的定律表明：**信息在过去和未来的点转换时，不会丢失，这被称为幺正性（unitarity）。**由于我们的量子门定义了状态的转变，因此它们也必须遵守幺正性。

其次，请注意，量子门是作用在量子比特上的。之前提到，量子比特实际上只是矢量，因此这意味着量子门必须以某种方式对矢量起作用。幸运的是，我们记得**矩阵实际上是向量的线性变换**！

结合这两种思想，我们将量子门视为幺正矩阵。幺正矩阵是复数的任何方阵，它的共轭转置等于它的逆。作为快速变换，可以通过取矩阵中每个元素的共轭（a + bi → a - bi），然后取矩阵的转置（元素ij→元素ji），可以找到矩阵的共轭转置。我们通常用“†”表示共轭转置。

关于幺正矩阵的一个重要性质是**范数（即向量的长度）不变性**。否则，一个量子比特被量子门处理过后，其概率和不为1！这没有任何意义，因为所有概率的总和必须始终等于1。

还应注意，根据定义，**幺正矩阵有逆**。这意味着我们不能将量子比特“分配”到任意状态。为了理解，我们假设有一个可以“**分配**”值的量子门，因此，可以将含有两个复数的任何矢量转换为含有两个复数的特定矢量。作为幺正矩阵，这个量子门具有的一些特定的表示，且该矩阵具有能够将特定矢量转换回操作前状态的逆矩阵！但是，在测量之前，量子比特可能处于任何状态，并且无法知道是哪个！因此，我们不能将量子比特“分配”到任意状态。在更高的层次上，所有量子门都是可逆的，这就是为什么我们经常**将量子计算视为可逆计算的一种形式。**

最后，请注意，因为我们的量子门是**幺正矩阵**，从定义上来说它们是**方阵**，所以我们的量子门必须**具有相等数量的输入和输出量子比特**（因为正方形矩阵将n个标准基向量映射到n列）！这与大多数逻辑门完全不同。例如，“与”门取两个输入并产生一个输出。

H和CNOT量子门

现在我们已经对我们将要做的事情有了一点了解了，现在以Hadamard门为例，矩阵H：

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

我们可以通过计算矩阵的共轭转置（H†）是否等于矩阵的逆（H-1）来确定H的幺正性：

$$HH^\dagger = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}^\dagger = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

另一个重要的量子门是“**控制非门**”，也称为**CNOT**。CNOT作用于两个量子比特，一个**控制量子比特**和一个**目标量子比特**。我们可以**将CNOT视为 “if语句”**——如果控制量子比特等于1，则CNOT将NOT（非门）应用于目标量子比特。因此CNOT又叫**控制非门。**

这是代表**CNOT**的矩阵。此矩阵将**控制量子比特**视为**右矢内的最右值**，将**目标量子比特**视为**最左值。**

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

将其作用到**|00**⟩上，得：

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} |00\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle$$

在此示例中，我们看到CNOT不会修改“00”的值，这是被预期的行为，因为CNOT仅在控制态为1时才反转目标态。

让我们来看看它对**|01**⟩的影响。

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} |01\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |11\rangle$$

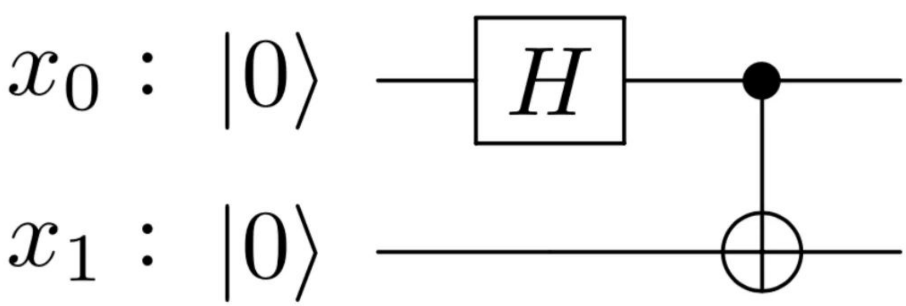
在这里，我们可以看到控制态等于1，因此CNOT发生概率反转。因此，结果是|11⟩。
尝试找出另外两种情况，|10⟩和|11⟩，你可以发现CNOT具有以下行为：

- `|00⟩ -> |00⟩`
- `|01⟩ -> |11⟩`
- `|10⟩ -> |10⟩`
- `|11⟩ -> |01⟩`

请注意，这恰恰是传统计算机中控制位为1时对目标位应用“**非**”的行为。
总结：**量子门是一个幺正矩阵，幺正性需得保证其概率之和为1，且量子计算机可逆。由于幺正矩阵是方阵，因此输入和输出的数目是一样的。我们了解到Hadamard和CNOT两个重要的量子门。**（还存在着更多的其他量子门。）

量子电路图

在已经了解量子比特和量子门的基础上，进一步学习**第一个量子电路图。**



量子电路图是我们对于量子程序的思考构建。将量子比特定义为行，从左到右应用量子门，构建量子电路图。遍历图的每一个部分。首先，我们有两个量子比特，每一行匹配一个量子比特，最上面的一行对应着x0的量子比特，下面对应着下x1的量子比特。将x0视为第0个量子比特，从0开始计数，标号x0：|0⟩和x1：|0⟩代表从状态|0⟩开始。

H是Hadamard门，应用于量子比特x0。●-⊕是CNOT门，●是**控制量子比特**，⊕是**目标量子比特**。“-”是为了帮助我们表示受影响的两个量子比特。换句话说，我们正在应用CNOT量子门，其中控制态为量子比特x0，目标态为x1。注意，我们这些门的应用顺序很重要。在此图中，我们首先应用H，然后应用CNOT。

翻译量子电路图

量子电路图只是量子编程的一种表达方式。它帮助我们了解量子计算机，但是其他的表示方法也是很有用的。我们能将我们的**图形式变为字符串符号的形式**，能够更好地帮助我们写作代码。以字符串表示能使之更轻松地转换为基础的数学，这个数学公式将告诉我们程序的预期输出。

首先，将图转换为字符串符号。我们使用**狄拉克表示法**，像写二进制数一样，第0个量子比特将是|00⟩中最右边的量子比特。这意味着量子比特x1是|00⟩中最左边的量子比特。（请注意，顺序的定义可以不同，但前后要保证统一。）

我们还需要转换“门”。

由于我们将H作用于量子比特x0而不作用任何东西到量子比特x1（等同于应用单位门，I），因此我们将其写为(I⊗H)。最后，我们翻译CNOT门，指定哪个量子比特是控制态，哪个是目标态。结果为CNOT [control = 0，target = 1]（I⊗H）|00⟩（注意，从右到左读取此字符串）。当编写将在量子计算机上运行的代码时，这些都将非常有用。

写出基础数学

写出了**量子电路图的字符串**表示形式之后就可以轻松地将我们的程序转换为基础**数学公式表达**的形式。可以分为三个部分，即CNOT [control = 0，target = 1]，(I⊗H)和|00⟩。每一部分都可以转换成矩阵，如下图的第一行所示：

$$\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}&\frac{1}{\sqrt{2}}&0&0\\\frac{1}{\sqrt{2}}&-\frac{1}{\sqrt{2}}&0&0\\0&0&\frac{1}{\sqrt{2}}&\frac{1}{\sqrt{2}}\\0&0&\frac{1}{\sqrt{2}}&-\frac{1}{\sqrt{2}}\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix}=$$

$$\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\\0\\0\end{pmatrix}=\begin{pmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{pmatrix}$$

我们甚至可以将矩阵相乘以找到结果状态向量，如上所示。该状态向量是量子计算完成后我们两个量子比特的预期状态。另外，我们可以将其视为程序的输出。它告诉我们每种可测量状态的概率幅度。

另外，还记得我们的混合态量子比特吗？请注意，我们无法再次编写量子比特x0和量子比特x1的纯态了，因为没有任何方法可以用张量积分解向量，因此我们的量子比特处于混合态！

测量状态向量

如果我们现在测量量子比特会怎么样？我们会收到什么结果？我们可以通过将状态向量分解为可测量状态来找出其结果。用标准基（也称为|0⟩和|1⟩）（我们也可以测量其他基数，但现在不用担心）来衡量量子比特。因此，我们两个量子比特系统的可测量状态为|00⟩、|01⟩、|10⟩和|11⟩。

$$\begin{pmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\0\end{pmatrix}+0\begin{pmatrix}0\\1\\0\\0\end{pmatrix}+0\begin{pmatrix}0\\0\\1\\0\end{pmatrix}+\frac{1}{\sqrt{2}}\begin{pmatrix}0\\0\\0\\1\end{pmatrix}=$$

$$\frac{1}{\sqrt{2}}|00\rangle+0|01\rangle+0|10\rangle+\frac{1}{\sqrt{2}}|11\rangle$$

就像我们可以使用|α|2来确定单个量子比特的|0⟩的概率一样，我们可以用同样的方法来确定测量值的概率。由于|01⟩和|10⟩的概率振幅为0，因此我们知道我们永远也不会测到该状态。并且我们将

以(1 / sqrt(2))^2 = 1/2的概率来测量到|00⟩和|11⟩。

现在，假设我们要将这两个量子比特分开很长的距离，然后再测量其中一个。在我们测量它的那一刻，我们也将知道另一个量子比特的值！这是因为我们知道量子比特只能是|00⟩或|11⟩。

这就是爱因斯坦所说的“**幽灵般的超距离作用**”，也称为**量子纠缠**（参见EPR佯谬）。

在量子计算机上运行

了解了量子比特、量子门和量子电路图的原理之后，让我们看看如何在真实的量子计算机上运行程序。可以使用Rigetti的量子计算机，因为他们目前向测试用户免费开放。我们也可以使用IBM的量子计算机。

这是Rigetti量子编程过程的基本概述：

1. 编写一个Python程序来指定您的量子电路和任何其他必需的代码；

2. 使用量子模拟器测试该Python程序；

3. 给Rigetti的量子计算机预留出时间；

4. 将您的程序发送到Rigetti的服务器；

5. 在Rigetti的服务器上执行程序（他们会将您的量子程序发送给他们的量子计算机）。

这是上面的量子电路图的Python版本：

```
from pyquil.quil import Program
from pyquil.api import *
from pyquil.gates import *

# Apply H to qubit 0, then CNOT to qubit 0 and 1
p = Program(H(0), CNOT(0, 1))
# Get info for a 2-qubit quantum virtual machine
qc = get_qc('2q-qvm')
# Simulate program
results = qc.run_and_measure(p, trials=10)
print(list(zip(results[0], results[1])))

# Apply H to qubit 1, then CNOT to qubit 1 and 2
p = Program(H(1), CNOT(1, 2))
# Get info for a real 2-qubit quantum computer named Aspen-4-2Q-A
qc = get_qc('Aspen-4-2Q-A')
# Send program to the quantum computer and run it
results = qc.run_and_measure(p, trials=10)
print(list(zip(results[1], results[2])))
```

结果是：

```
[(0, 0), (1, 1), (1, 1), (0, 0), (0, 0), (0, 0), (1, 1), (0, 0), (0, 0)
[(0, 0), (0, 1), (1, 1), (1, 1), (1, 1), (0, 0), (0, 0), (1, 1), (1, 0)
```

第一行对应于模拟器，结果似乎合理——得到的[0, 0]与[1, 1]各占一半。但是，使用真实的量子计算机，我们还收到不可能存在的[0, 1]和[1, 0]。这是怎么回事？

问题在于，如今（2019年）的量子计算机仍然很容易出错。例如，当尝试将量子比特初始化为0时，会有2-3%的错误率。每个量子比特门操作的错误率为1-2%，而两个量子比特门操作的错误率约为3-4%。甚至在测量量子比特时也会有错误率！这些错误会累积并导致错误的结果。

结尾

在本文中，我们可以知道，尽管错误率很高，但量子计算机实际上确实存在并且可以正常工作。尽管因不同公司的物理实现有所差别，对其编程的许多概念却保持一致。

我们认为量子比特是具有单位长度的两个复数的向量，并将量子门视为可逆的幺正矩阵。量子计算是概率性的，因为一旦测量，两个相同的量子比特可能具有不同的值。在较高的层次上，我们可以将量子编程视为对复数应用的线性代数。

我们使用量子电路图来表示我们的量子程序，然后将其转换为Python以在真实的量子计算机上运行。

至此，希望你已经可以写出自己的量子程序了。

参考文献：

[1] L. Susskind, Lecture 1 Quantum Entanglements, Part 1 (2008)

[2] Basis vector ordering in Qiskit (2019), Qiskit

[3] R. Smith, Someone Shouts “01000”！Who is excited? (2017), arxiv

[4] Qubit Quality (2019), Quantum Computing Report

原文链接：towardsdatascience.com/...

发布于 2019-11-27

量子计算机 量子计算 量子力学

张ben 2019-11-28

只看懂一半,能不能再简单一点?

影月 2019-11-28

看了本源量子的编程入门。

四眼老胖 2019-12-03

院爷...求科普......普、普通的普

光影帽子 2019-12-11

感觉就是矩阵变换.....不知道这种量子电路有没有其他的意义呢?

霄池 回复 光影帽子 05-09

如果有n位量子位，就有2^n个系数（振幅），计算能力是指数增长的。

但如果你想算一个nxn的矩阵，那计算量也是指数增长的。

hdyjzdj 2019-12-19

我竟然看懂了，哈哈

不知者无罪 2019-12-31

没有一点计算机语言基础的怎么学啊[飙泪笑]

喜欢他 03-31

弱弱的问一下。不知道可不可以请求写一个量子编程的量子测量问题(魏格纳朋友实验)，当然不是白白写的，我要愁死了，