Quantum circuits, registers and the language QUANT

Introduction

Roughly speaking, a **quantum computer** is a computer that is able to use certain capabilities based on quantum physics in addition to the usual ("classical") capabilities that computers have.

The idea to use quantum physics for computation arose in the 1980s with work by the physicists Richard Feynman and David Deutsch. Interest increased considerably in the 1990s when Peter Shor gave an efficient quantum algorithm for integer factorisation. It had been assumed that integer factorisation was difficult; a fast factorisation algorithm could be used to break the RSA public-key cryptosystem. RSA was the first public-key cryptosystem to be published and is still the most widely used, underpinning the security of a large proportion of modern electronic communications[6]. Since then, many quantum algorithms have appeared, some improving considerably on the best classical algorithms. But, to have any impact in practice, they will have to be implemented on actual quantum computers and applied to large inputs.

Several dozen quantum computers have been built. Many of these are experimental, while some have been used for highly specialised applications. But they are nowhere near powerful enough or robust enough to be useful on a large scale. Practical quantum computing still faces huge technical challenges including adequately protecting the delicate computations from being perturbed by the surrounding environment. In spite of this, there is considerable optimism about their future, and a widespread belief that, in time, their effect will be revolutionary (as well as some scepticism about this). Although they cannot yet be used to break RSA, the threat they pose has been taken very seriously by cryptographers, whose work has led to the development of new quantum-resistant cryptosystems: `https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms`.

For a recent review of the field's current progress and prospects, see [1].

Mathematically, the heart of a quantum computer has three parts:

- a *register* that stores information, and in fact can store multiple data items simultaneously, in a *superposition*, although these data items cannot be accessed in standard classical ways;
- a *quantum circuit expression*, in which certain basic components (called *gates*) are combined in order to produce a transformation that is applied to the register to produce another register;
- *measurement*, in which some part of a register is measured (i.e., read), but the result is determined probabilistically by the entire contents of the register, and the act of measurement reduces the amount of information held in the register.

In this assignment, we just consider registers and quantum circuit expressions, and in fact we only consider restricted types of these. Nonetheless, our registers and expressions are sufficient, in principle, to represent the pre-measurement parts of quantum computations arbitrarily accurately. We define these concepts now.

A **register** is a 2^n-dimensional vector of unit length. It may contain complex numbers (e.g., $i = \sqrt{-1}$). Examples of registers include:

$$n = 0: \quad \begin{pmatrix} 1 \end{pmatrix} \qquad n = 0: \quad \begin{pmatrix} -i \end{pmatrix} \qquad n = 1: \quad \begin{pmatrix} 0.6i \\ -0.8 \end{pmatrix}$$

$$n = 2: \quad \begin{pmatrix} 1/2 \\ -1/2 \\ -1/2 \\ 1/2 \end{pmatrix} \qquad n = 3: \quad \begin{pmatrix} 0.6 \\ 0.2 \\ -0.2 \\ -0.4 \\ 0.2 \\ -0.2 \\ 0.4 \\ -0.4 \end{pmatrix}$$

[6] Shor also gave an efficient quantum algorithm for another number-theoretic problem called discrete log. A fast discrete log algorithm would break the Diffie-Hellman key-exchange scheme, another important cryptographic tool.

There are two particular registers with $n = 1$ that we use repeatedly:

traditional name	our name	vector
$\|0\rangle$	`k0`	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$
$\|1\rangle$	`k1`	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

To define quantum circuit expressions, we need two different ways of combining matrices: ordinary matrix multiplication, and another operation called *Kronecker product* which is probably new to you. We discuss these in turn.

Ordinarily, multiplying two matrices A and B is only defined if the number of columns of A equals the number of rows of B. In this assignment, we introduce a new **error matrix** whose sole role is to indicate that an error has been made, at some stage in a calculation, due to trying to multiply incompatible matrices. It has no rows, columns or entries, and we regard it as having $n = -1$. Once an error matrix appears in a calculation, you cannot get rid of it: the result of any subsequent calculation with it will again be the error matrix. With this little "hack", we can allow *any* two matrices to be multiplied together; we just have to keep in mind the possibility of producing the error matrix and having it "swamp" the remainder of our current calculation.

We now define the Kronecker product.

Let $A = (a_{ij})_{r\times c}$ be an $r \times c$ matrix, with r rows and c columns, whose i, j-entry is a_{ij}. Similarly, let $B = (b_{kl})_{s\times d}$ be an $s \times d$ matrix, with s rows and d columns, whose k, l-entry is b_{kl}. Then the **Kronecker product** $A \otimes B$ is the $rs \times cd$ matrix, with rs rows and cd columns, formed by multiplying each entry of A by a copy of B:

$$A \otimes B \quad = \quad \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1c}B \\ a_{21}B & a_{22}B & \cdots & a_{2c}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1}B & a_{r2}B & \cdots & a_{rc}B \end{pmatrix}$$

It can be shown that the entry in row $(i-1)s+k$ and column $(j-1)d+l$ of $A \otimes B$ is $a_{ij}b_{kl}$ (where $1 \le i \le r$, $1 \le j \le c$, $1 \le k \le s$, $1 \le l \le d$). For example, if both A and B are 2×2 matrices, with

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \qquad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix},$$

then

$$A \otimes B = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{pmatrix}.$$

Some concrete examples:

Let I be the usual 2×2 identity matrix,

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{1}$$

Define H by

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}. \tag{2}$$

Then

$$I \otimes I \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad H \otimes H \;=\; \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{pmatrix},$$

$$I \otimes H \;=\; \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \qquad H \otimes I \;=\; \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}. \quad (3)$$

The Kronecker product can also be applied to our vectors, which are, after all, just matrices with only one column:

$$\texttt{k0} \otimes \texttt{k1} \;=\; \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \qquad \texttt{k1} \otimes \texttt{k0} \;=\; \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

We see from these examples that Kronecker product is not commutative in general.

Note that the Kronecker product is *always* defined, regardless of the dimensions of the matrices.

We also use the **Kronecker power**, which just means taking the Kronecker product of some number of copies of a matrix. Mathematically, the n-th Kronecker power of a matrix A is usually written $A^{\otimes n}$. In our quantum expressions (to be defined soon), we represent this as $\mathrm{KronPow}(A, n)$. So we have

$$\mathrm{KronPow}(A, n) \;=\; A^{\otimes n} \;=\; \underbrace{A \otimes A \otimes \cdots \otimes A}_{n \text{ copies of } A}.$$

For example,

$$\mathrm{KronPow}(H, 2) \;=\; H \otimes H \;=\; \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

Quantum gates

We use a small set of fundamental matrices, called **quantum gates**, to help build our expressions. The quantum gates we use are:

$$I \;=\; \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad H \;=\; \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \quad X \;=\; \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y \;=\; \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z \;=\; \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

$$\mathrm{CNOT} \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \qquad \mathrm{TOF} \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

This set of gates is sufficient to build a quantum circuit to approximate any quantum computation arbitrarily closely. In fact, even the two-gate set $\{H, \text{TOF}\}$ is sufficient for that.

H is known as the **Hadamard gate**, and X, Y and Z are the **Pauli-X**, **Pauli-Y** and **Pauli-Z gates**. CNOT is the **controlled-not gate** and TOF is the **Toffoli gate**. The matrix Y includes the complex numbers $\pm i$, where $i = \sqrt{-1}$. The other matrices are all real matrices (although combining them with Y in any way will usually produce some complex numbers of the form $a + bi$ where a and b are real).

Languages for quantum expressions

Quantum circuit expressions are defined inductively as follows.

- Each of `I`, `H`, `X`, `Y`, `Z`, `CNOT` and `TOF` is a quantum circuit expression.
- If Q is a quantum circuit expression, then so is (Q).
- If P and Q are quantum circuit expressions, then so is $P * Q$. (This represents ordinary matrix multiplication.)
- If P and Q are quantum circuit expressions, then so is $P \otimes Q$. (This represents Kronecker product.)
- If Q is a quantum circuit expression and n is a nonnegative integer, then $\text{KronPow}(Q, n)$ is a quantum circuit expression. (This represents Kronecker power.)

This inductive definition can be used as the starting point for writing a Context-Free Grammar for these expressions.

Quantum register expressions are defined inductively as follows.

- Each of `k0` and `k1` is a quantum register expression.
- If R is a quantum register expression, then so is (R).
- If R and S are quantum register expressions, then so is $R \otimes S$.
- If R is a quantum register expression and n is a nonnegative integer, then $\text{KronPow}(R, n)$ is a quantum register expression.
- If Q is a quantum circuit expression and R is a quantum register expression, then $Q * R$ is a quantum register expression.

This inductive definition can also be used as the starting point for writing a Context-Free Grammar.

The languages QCIRC, QREG and QUANT

We define three languages to describe the types of quantum expressions we are working with.

- QCIRC is the language, over the fourteen-symbol alphabet {`I`, `H`, `X`, `Y`, `Z`, `CNOT`, `TOF`, `*`, $\otimes$, `NUMBER`, KronPow, `(`, `)`, `,` }, of valid quantum circuit expressions.
- QREG is the language, over the sixteen-symbol alphabet {`I`, `H`, `X`, `Y`, `Z`, `CNOT`, `TOF`, `k0`, `k1`, `*`, $\otimes$, `NUMBER`, KronPow, `(`, `)`, `,` }, of valid quantum register expressions.
- QUANT is their union: QUANT = QCIRC $\cup$ QREG.

When representing members of these languages in text, we replace $\otimes$ by the three-letter string `(x)`, which is intended to be as close as we can get, with keyboard characters, to a cross in a circle! (Note that it uses *lower-case* `x`.) Always remember that it is our text representation of the Kronecker product symbol $\otimes$; it is *not* an expression x enclosed in parentheses, and it is *not* an argument x being supplied to some function! In this assignment, we *only* use the lower-case letter `x` in this way, enclosed in *one* pair of parentheses in order to represent $\otimes$.

In lexical analysis, we treat `(x)` as the sole lexeme associated with the token `KRONPROD`, and we treat `KronPow` as the sole lexeme associated with the token `KRONPOW`. We also treat `k0` and `k1` as the sole lexemes associated with tokens `KET0` and `KET1`, respectively. This follows a widespread convention that token names are upper-case.

`I`, `H`, `X`, `Y`, `Z`, `CNOT`, `TOF` are tokens representing the names of the matrices we use as quantum gates. Each is also the sole lexeme for its token.

The following table gives some members of QUANT. For each, we indicate in the right column whether it belongs to QCIRC or QREG.

quantum expression	string representation	evaluates to	comments
I	`I`	see (1)	QCIRC: ✓ QREG: ✗ This is a 2×2 identity matrix and is a valid quantum circuit expression. But it's not a vector, so is not a quantum register expression.
$H \otimes I$	`H (x) I`	see (3)	QCIRC: ✓ QREG: ✗ This is also a valid quantum circuit expression, but is not a vector.
$\texttt{k0} \otimes \texttt{k1}$	`k0 (x) k1`	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$	QCIRC: ✗ QREG: ✓
$\text{KronPow}(\texttt{k1}, 2)$	$\text{KronPow}(\texttt{k1}, 2)$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	QCIRC: ✗ QREG: ✓
$H * \texttt{k0}$	`H*k0`	$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$	QCIRC: ✗ QREG: ✓
$(H * \texttt{k0}) \otimes (H * \texttt{k1})$	`(H*k0) (x) (H*k1)`	$\begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}$	QCIRC: ✗ QREG: ✓
$(H \otimes H) * (\texttt{k0} \otimes \texttt{k1})$	`(H (x) H) * (k0 (x) k1)`	$\begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}$	QCIRC: ✗ QREG: ✓
$CNOT * \texttt{k1}$	`CNOT*k1`	error	QCIRC: ✗ QREG: ✓ The attempt to multiply 4×4 matrix CNOT by 2-element column vector `k1` results in error. But the string still belongs to QREG.
$(I \otimes I) * H * \texttt{k1}$	`(I (x) I)*H*k1`	error	QCIRC: ✗ QREG: ✓ Again, an attempt at an illegal matrix multiplication results in error. But the string still belongs to QREG.

Some examples of *invalid* quantum expressions (i.e., not members of QUANT):

expression	text repr'n	comment
$H \otimes$ `k0`	`H (x) k0`	The inductive definition of a QUANT does not allow for the Kronecker product of a quantum circuit expression and a quantum register expression. (Mathematically, such a Kronecker product gives a valid matrix, but not one that represents either a quantum circuit expression or a quantum register expression.)
`k0` $* H$	`k0*H`	This product is the wrong way round. As it is, it's a 2×4 matrix, so is neither a quantum circuit expression nor a quantum register expression. For the other way round, `H*k0`, see previous table.
$H + I$	`H+I`	$+$ is not a valid operation in QUANT.
KronPow$(X, -1)$	KronPow$(X, -1)$	negative exponents in KronPow are not valid in QUANT.
`Power`$(H, 2)$	`Power(H,2)`	`Power` is not valid in QUANT; we only use it in PLUS-TIMES-POWER.

Quantum computation

The previous sections give as much detail as you need to know, at a minimum, to do the assignment. In this section, we give a very brief introduction to *some* of the concepts of quantum computing. It won't be enough to enable you to go away and write quantum algorithms. But, if you do want to learn how to do that, what you have learned from reading this section and doing this assignment will make it a bit easier to get started.

Let's consider *quantum registers* again. A quantum register has 2^n entries, for some nonnegative integer n. We could index them by the numbers $0, 1, \ldots, 2^n - 1$. We could also index them by strings of n bits, with these strings being the n-bit binary representations of those index numbers (with leading zeros allowed, this time). The following table shows a register with $n = 2$ and four entries $\frac{-1}{\sqrt{2}}, 0, \frac{-1}{2}, \frac{1}{2}$. The register is shown as a four-element vector on the right. You can verify that its length is $(|\frac{-1}{\sqrt{2}}|^2 + 0^2 + |\frac{-1}{2}|^2 + |\frac{1}{2}|^2)^{1/2} = \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{4}\right)^{1/2} = 1^{1/2} = 1$, as required. On the left, we give it in tabular form. The register's entries are in the third column (amplitude), with the first and second columns showing its indices as numbers and bit-strings respectively.

outcome		amplitude	probability
0	`00`	$-\frac{1}{\sqrt{2}}$	$\frac{1}{2}$
1	`01`	0	0
2	`10`	$-\frac{1}{2}$	$\frac{1}{4}$
3	`11`	$\frac{1}{2}$	$\frac{1}{4}$

register as vector

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

The bit-strings that index the entries of the quantum register are its **outcomes**. The entries themselves are called the **amplitudes** of the outcomes.[7] So, in the above register, the outcome `00` has amplitude $\frac{-1}{\sqrt{2}}$, while outcome `11` has amplitude $\frac{1}{2}$.

[7] We follow Lipton and Regan [5] in moving freely between (i) the index of an entry in a vector representing a register and (ii) a vector that has 1 at that index and 0 elsewhere.

A classical register, in a classical computer, stores just *one* bit-string. But quantum registers are very different. All the 2^n outcomes, of n bits each, exist *simultaneously* in some sense, each with its own amplitude. Outcomes of zero amplitude (like `01` in the above example) are impossible, but all outcomes with nonzero amplitude are present to some extent. We say that the register is in a **superposition** of all its possible outcomes.

In general, the amplitudes may be arbitrary complex numbers of length ≤ 1, subject to the constraint that the sum of the squares of their absolute values is 1 (so that, as a vector of 2^n entries, its length is 1).

A classical register has n positions (for some n). Each position holds one bit, either `0` or `1` (but not both!).

A quantum register also has n positions (for some n), but because of superposition, the information at that position is not just one single bit. Rather, it is a superposition of bits, this superposition being induced by the register's superposition of outcomes. This superposition of bits, in a particular position, is called a **qubit**, and we say that the register has n **qubits**.

Mathematically, classical registers may be regarded as a special case of quantum registers. A classical register containing a given bit-string is just a quantum register in which that bit-string, as an outcome, has amplitude 1 and all other outcomes have amplitude 0. For example, a two-bit classical register containing `01` is equivalent to the quantum register $\texttt{k0} \otimes \texttt{k1}$. Such a register is called a **basis register**.

We mentioned *measurement* on page 12 in our summary of the heart of a quantum computer. We don't use it in this assignment, but it's the only way we get output from a quantum computer, so let's consider it now.

In classical computing, reading a register will recover whatever bit-string is stored in it at the time, and nothing else. Given that a quantum register contains 2^n outcomes *at once* (in a superposition), we might hope to get much more out of it! But it's a bit trickier than that. When we read a quantum register, we are sampling from a *probability distribution* over its outcomes. Each outcome has a probability of being chosen, with that probability being the square of the absolute value of its amplitude. These probabilities do indeed sum to 1, because the length of the register (as a vector) is 1.

The probabilities for each of the outcomes in our example register above are given in the fourth column of the table. If we read this register, we have a probability of $\frac{1}{2}$ of getting `00`. Outcomes `10` and `11` each have a probability of $\frac{1}{4}$, while outcome `01` is impossible.

For reasons related to the underlying physics, the act of reading a register is called **measurement**.

We have so far envisaged measuring (i.e., reading) an entire quantum register. But it is also possible to measure individual qubits, i.e., to read what is at a given position. Suppose we wanted to measure the second qubit in the two-qubit quantum register given above. The probability of it being `0` is the sum of the squares of the absolute values of the amplitudes of those outcomes with second bit 0:

$$\begin{aligned} \text{Pr(measurement of 2nd bit gives } \texttt{0}) &= |\text{amplitude of } \texttt{00}|^2 + |\text{amplitude of } \texttt{10}|^2 \\ &= |\tfrac{-1}{\sqrt{2}}|^2 + |\tfrac{-1}{2}|^2 \\ &= \tfrac{3}{4}. \end{aligned}$$

Similarly,

$$\begin{aligned} \text{Pr(measurement of 2nd bit gives } \texttt{1}) &= |\text{amplitude of } \texttt{01}|^2 + |\text{amplitude of } \texttt{11}|^2 \\ &= |0|^2 + |\tfrac{1}{2}|^2 \\ &= \tfrac{1}{4}. \end{aligned}$$

A quantum computation proceeds by

- starting with a register with a single outcome having amplitude 1 and all other outcomes having amplitude 0 (such a register can be formed from a Kronecker product of copies of `k0` and `k1`);

- applying a quantum circuit — which we model as a quantum circuit expression — to the register, thereby producing a new register;
- measuring one or more qubits of this new register. This becomes the output of the computation.

The initial register is *not* the means of providing input to the quantum computer; it is more analogous to the Start State of an automaton. Input is provided to the quantum computer by using the input in the construction of the quantum circuit. So, that circuit is determined in some computable way by the input. Specification of how the circuit is computed from the input is part of the specification of a quantum algorithm.

Let's look at a couple of very simple quantum computations.

Suppose we have $n = 1$ (one qubit) and start with $\texttt{k0}$. Let's use a quantum circuit consisting solely of H. Then the computation produces

$$H * \texttt{k0} \;=\; \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \;=\; \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}.$$

Then, if we measure the register, we obtain

$$\text{outcome} \;=\; \begin{cases} 0, & \text{with probability } \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}; \\ 1, & \text{with probability } \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}. \end{cases}$$

So this is like tossing a fair coin.

Now suppose we use a one-qubit circuit consisting of two consecutive applications of H. So the quantum circuit expression is now $H * H$. Starting again with $\texttt{k0}$, we obtain

$$H*H*\texttt{k0} \;=\; \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \;=\; \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \;=\; \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

So, this time, we just end up with $\texttt{k0}$ again which has only one possible outcome. Mathematically, this is no surprise because the matrix H is its own inverse: $H * H = I$. Let's consider what happens in more physical terms:

1. one application of H takes us from $\texttt{k0}$, which has only one possible outcome, to a superposition of two outcomes (so H seems to "spread out" the available amplitude across more outcomes);
2. then another application of H — which we might intuitively expect to "spread things out" even further — actually creates "cancellation" so that only one outcome is left standing. This phenomenon is known as **interference**.

Now let's graduate to two qubits and use the quantum circuit $\text{CNOT}*(H \otimes I)$ with initial register $\texttt{k0} \otimes \texttt{k0}$. The final register is

$$\text{CNOT}*(H\otimes I)*(\texttt{k0}\otimes\texttt{k0}) \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \;=\; \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}.$$

If we measure the final register, it will give outcome $\texttt{00}$ with probability $\frac{1}{2}$ and outcome $\texttt{11}$ with probability $\frac{1}{2}$. Note that, in both these possible outcomes, the left and right bits are identical. So, in the register, the left and right qubits are not independent; in fact, they are **entangled**.