Quantum Algorithms via Linear Algebra

— Essential Basics from Linear Algebra

Dao-Yun Xu

College of Computer Science and Technology, Guizhou University

Quantum Algorithms via Linear Algebra

A PRIMER

RICHARD J. LIPTON AND KENNETH W. REGAN

“Quantum Algorithms via Linear Algebra”

The book has following characterizations:

(1) The quantum computation can be described in terms of elementary linear algebra, without background in quantum theory, quantum mechanics, or quantum anything.

(2) It is an introduction to quantum algorithms unlike any other. It is short, yet it is comprehensive and covers the most important and famous quantum algorithms.

(3) It assumes minimal mathematically background, yet it explains quantum algorithms, yet steers clear of the notorious philosophical problems and issues of quantum mechanics.

Our goal is to explain quantum algorithms via linear algebra.

The mathematics that is required to understand them is linear algebra: vectors, matrices, and their basic properties.

That is all. So these are really linear-algebraic algorithms.

We will represent the operations by matrices, and we give several equivalent stipulations for **unitary matrices**.

Our model's programs will simply be compositions of unitary matrices.

Outline

1. Introduction

2. Numbers and Strings

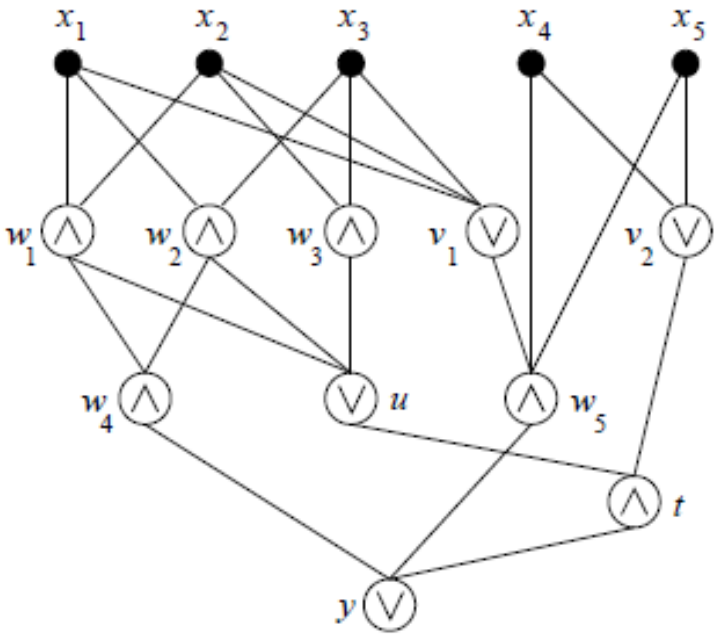
3. Basic Linear Algebra

1. Introduction

Classical computers–Boolean circuits:

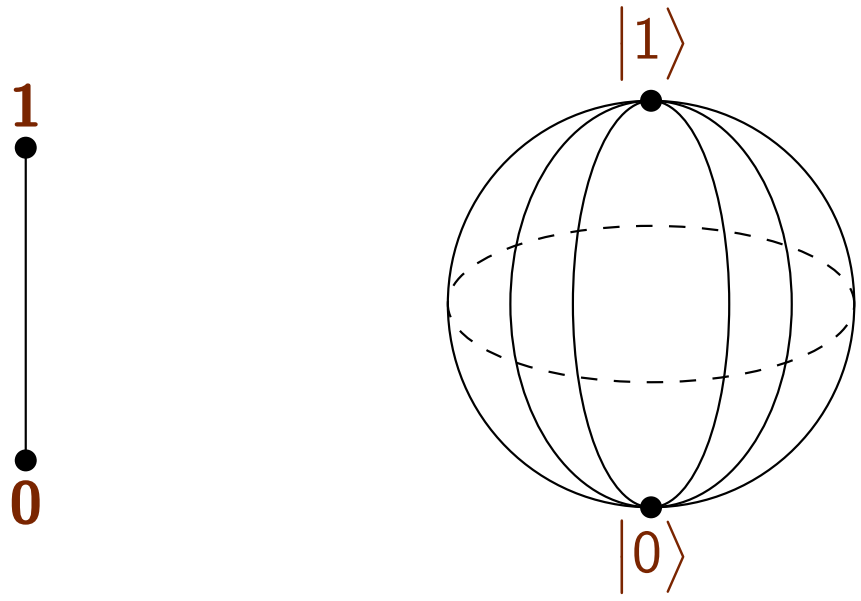
The classical computers are from **integrated Boolean circuits**, which information is manipulated as **bits**, 0 or 1.

The string function is defined as $f : \{0,1\}^n \to \{0,1\}^m$.



Quantum computers–quantum circuits:

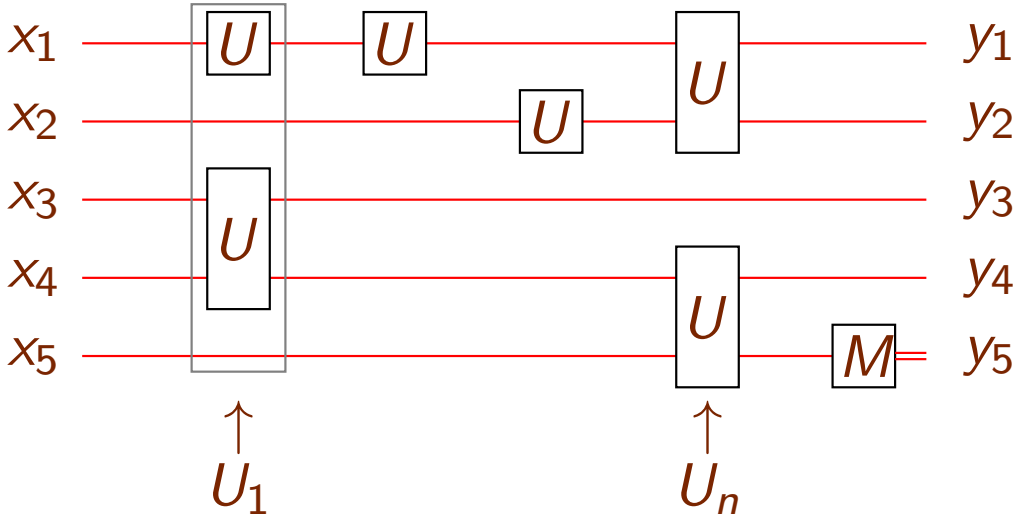
The quantum computation operates on **qubits**, which are based on complex-number values, **not just 0 and 1**.



States spaces as convex sets:

left: one classical bit; *right:* one quantum bit (qubit).

$$0 \leftrightarrow e_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad 1 \leftrightarrow e_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ (one basic qubit)}$$



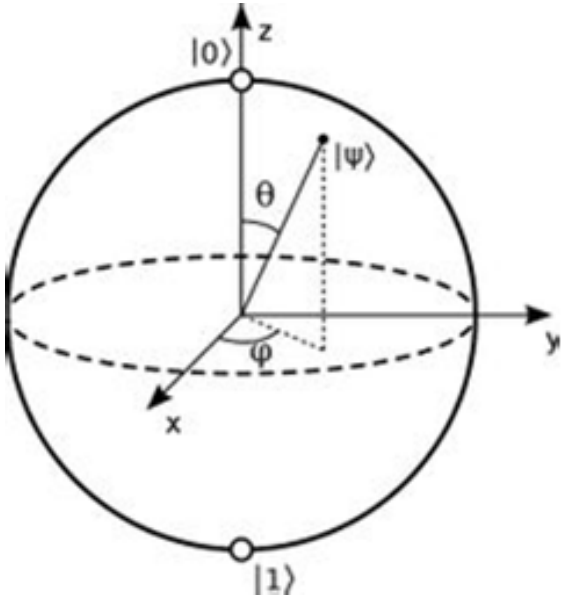
$$U = U_n U_{n-1} \cdots U_1 \qquad y = Ux \ (N = 2^5).$$

They can be read only by measuring, and the readout is in "classical bits"

1.1. Unitary transformation in Hibert (Sphere) Spaces

In quantum algorithms, we need only a simple model—Unitary transformation on **Ball Sphere** in Hibert Spaces.

$$\mathbb{S} = \{x \in \mathbb{C}^N \mid ||a||_2 = 1\} \ (N = 2^n).$$



Bloch ball sphere

1.2. The Space and the States

States: The state is a point on the ball sphere $\mathbb{S}$, described by a single unit vector $\mathbf{a}$ in some fixed vector space of dimension $N = 2^n$.

$$\mathbf{a} = \begin{bmatrix} a_0 \\ \vdots \\ a_{N-1} \end{bmatrix} = \left|a\right\rangle \text{ (Dirac notations)}$$

where a_k is a real or complex number, $\sum_{k=0}^{N-1} |a_k|^2 = 1$.

Each entry is called an **amplitude**, and is corresponded to a probability a_k^2.

A state is called **pure** if it cannot be written as a convex combination $\lambda a + (1-\lambda)b$ of other states.

Basis states: The basis states are denoted by the vectors e_k whose entries are 0 except for a 1 in place k.

We identify e_k with the index k itself in $[0, N-1]$ and then further identify k with the k-th string x in a fixed total ordering of $\{0,1\}^n$, i.e., $e_k = e_{x_1 \cdots x_n}$.

The basis states correspond to all the length-n binary strings.

$$e_{x_1 \cdots x_n} \leftrightarrow x_1 \cdots x_n \leftrightarrow \left| x_1 \cdots x_n \right\rangle$$

General states: A N-vector $\mathbf{a}$ that is not a basis state is a **superposition**. *Two or more basis states have nonzero amplitude in any superposition, and only one of them can be observed individually in any measurement*.

Representation and Measuring

Representation and Measuring of vectors:

Let V be n-dimension **Hilbert space** over C (complex field) with orthogonal basis $e_1, \cdots e_n$, where the norm $||e_i||_2 = 1$. Thus a vector α in V is represented as

$$\alpha = a_1 e_1 + \cdots + a_n e_n$$

where (complex) vector $(a_1, \cdots, a_n)^\top$ is called **coordinate** with respect to $\{e_1, \cdots e_n\}$.

Inner product $< \alpha, e_i >= a_i$ is viewed as **measuring amplitude** to α by e_i. We also view $|a_n|^2$ as the **probability** of the vector α can be measured by e_i.

Measuring in probability

Examples:

$$a = a_{00}e_{00} + a_{01}e_{01} + a_{10}e_{10} + a_{11}e_{11}$$

Getting state e_{00} by measuring in probability $|a_{00}|^2$.

Getting state $\frac{1}{\sqrt{|a_{00}|^2+|a_{11}|^2}}(a_{00}e_{00} + a_{11}e_{11})$ in probability:

$$|a_{00}|^2 + |a_{11}|^2.$$

Representation and Measuring of functions:

Assume that an integral function $f(x)$ is defined in the interval $[a, b]$, and the collection $\{\varphi_0, \varphi_1, \cdots, \varphi_n, \cdots\}$ of functions is a complete orthogonal. Thus, the function is represented as

$$f(x) \sim a_0\varphi_0 + a_1\varphi_1 + \cdots + a_n\varphi_n + \cdots$$

where $a_n = \frac{1}{Z}\int_a^b f(x)\varphi_n dx$ is called **coordinate** with respect to φ_n, $Z^2 = \int_a^b f^2(x)dx = \sum_{n\geq 0} a_n^2$.

Inner product $< f, \varphi >= \frac{1}{Z}\int_a^b f(x)\varphi_n dx$ is viewed as **measuring** to f by φ_n. We also view a_n^2 as the probability of the function f can be measured by φ_n.

1.3 The Operations—Unitary matrices

The operations are required with properties from (five) hypothesises of quantum mechanics:

(1) It must **map any point on the unit sphere to some (other) point on the unit sphere**.

(2) It is **linear**.

(3) It must also be **invertible**.

Operations with all these properties are called **unitary**.

We will represent the operations by matrices, and we give several equivalent stipulations for **unitary matrices**.

Our model's programs will simply be compositions of unitary matrices.

In the model we must understand what state the computation starts in, how it moves from one state to another, and how we get information out of the computation.

Start: We will assume that the start state is always the elementary vector

$$\mathbf{e_0} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

of length N.

Move: If the system is in some state a, then we can move it by applying a unitary transformation U. a will move to b where $b = Ua$. Note that if a is a unit vector, then so is b.

End: We get information out of the quantum computation by making a measurement. If the final state is c, then k is seen with probability $|c_k|^2$.

Often we will have a distinguished set S of indices that stand for *accept*, with outcomes in $[0, N-1] \setminus S$ standing for **reject**.

1.4. Input and Output

In classical computations, we are allowed to change the program each time in different input to solve a different problem.

Consider a program of this form:

$M = 21$; (input)

$x = Factor(M)$;

$procedure\ Factor(z)\ \{...\}$

In a classical deterministic model of computation, every computation has one, definite, output. Even if the model is randomized, every random input still determines a single computation path, whose output is definite before its last step is executed.

In quantum computation, this integration of inputs with programs is more characteristic of quantum than classical computation.

Think of the transformation U as the program, so varying U is exactly the same as varying the above program in the classical case.

$$U = U_n U_{n-1} \cdots U_1$$

A quantum computation presents the user at the end with a slot machine. Measuring is pulling the lever to see what output the wheels give you.

You control how the machine is built, and hopefully you've designed it to make the probabilities work in your favor.

As with the input, however, the focus shifts from a given binary string to the machinery itself, and to the action of **sampling a distribution** that pulling the measurement lever gives you.

Outline

1. Introduction

2. Numbers and Strings

3. Basic Linear Algebra

2. Numbers and Strings

Boolean string

A Boolean string is a string that consists solely of bits: a bit is either 0 or 1.

(1) If m is a natural number, then it can uniquely be written as a binary number:

$$m = 2^{n-1}x_{n-1} + \cdots + 2x_1 + x_0,$$

where each x_i is a bit, and we insist that x_{n-1} is nonzero. Then we can use m to denote the Boolean string $x_{n-1} \cdots x_1 x_0$.

(2) **Boolean inner product:** $x_1 y_1 \oplus \cdots \oplus x_m y_m$.

(3) A Boolean strings can represent subsets of a set.

If the set is $\{1, 2, 3\}$, in that order, then 000 corresponds to the empty set,011 to $\{2, 3\}$, and so on.

Asymptotic Notation

- $s(n) = O(t(n))$: if there are constants c, d such that for all n,
$$s(n) \leq c \cdot t(n) + d.$$
- $s(n) = \Omega(t(n))$ if $t(n) = O(s(n))$, and
- $s(n) = \Theta(t(n))$ if $s(n) = O(t(n))$ and $s(n) = \Omega(t(n))$.

Outline

1. Introduction

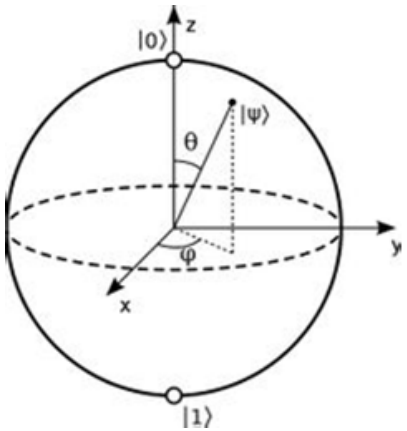
2. Numbers and Strings

3. Basic Linear Algebra

3. Basic Linear Algebra

3.1. Hilbert Spaces: A real Hilbert space is the usual Euclidean space. We will use $\mathbb{H}_N$ to denote this space of dimension N. The elements are real vectors a of dimension N. They are added and multiplied by scalars in the usual manner. In a Hilbert space, each vector a has a *norm* $||a|| = (\sum_{k=1}^{N} |a_k|^2)^{1/2}$.

A **unit vector** is just a vector of norm 1. Unit vectors together comprise the **unit sphere** in any Hilbert space.



Bloch ball sphere: $\left|\psi\right\rangle = cos(\frac{\theta}{2})\left|0\right\rangle + e^{i\varphi} sin(\frac{\theta}{2})\left|1\right\rangle \leftrightarrow (\theta, \varphi)$

3.2. Products and Tensor product

Products: The ordinary **Cartesian product** of an m-dimensional Hilbert space $\mathbb{H}_1$ and an n-dimensional Hilbert space $\mathbb{H}_2$ is the $(m+n)$-dimensional Hilbert space obtained by **concatenating vectors** from the former with vectors from the latter. Its vectors have the form $a(i)$ with $i \in [m+n]$.

Tensor product: The tensor product $\mathbb{H}_1 \otimes \mathbb{H}_2$ has vectors of the form $\mathbf{a}(k)$, where $1 \leq k \leq mn$. Indeed, k is in $1-1$ correspondence with pairs (i,j) of indices where $i \in [m]$ and $j \in [n]$. Because we regard indices as strings, we can write them juxtaposed in **lexicographic order** as $a(ij)$.

The tensor product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is the vector $\mathbf{c} = \mathbf{a} \otimes \mathbf{b}$ defined by $\mathbf{c}(ij) = \mathbf{a}(i)\mathbf{b}(j)$.

$$\begin{bmatrix} a_1 \\ a_2 \\ a_2 \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ a_1 b_2 \\ a_1 b_3 \\ \vdots \\ a_3 b_1 \\ a_3 b_2 \\ a_3 b_3 \end{bmatrix} = \begin{bmatrix} c_{11} \\ c_{12} \\ c_{13} \\ \vdots \\ c_{31} \\ c_{32} \\ c_{33} \end{bmatrix}$$

The tensor product of two matrices: $A \otimes B = (a_{ij} B)$.

Separable quantum state: A vector c denoting a pure quantum state is **separable** if it is the tensor product of two other vectors, i.e., it is of form:

$$c = a \otimes b$$

otherwise it is **entangled**.

The vectors e_{00} and e_{11} are separable, but their unit-scaled sum $\frac{1}{\sqrt{2}}(e_{00} + e_{11})$ is entangled.

The standard basis vectors of $\mathbb{H}_1 \otimes \mathbb{H}_2$ are separable, but this is not true of many of their linear combinations, all of which still belong to $\mathbb{H}_1 \otimes \mathbb{H}_2$ because it is a Hilbert space.

Often our quantum algorithms will operate on the product of two Hilbert spaces, each using binary strings as indices, which gives us the space of vectors of the form $a(xy)$.

Here x ranges over the indices of the first space $\mathbb{H}_1$ and y over those of the second space $\mathbb{H}_2$. Note that the writing $a(xy)$ does not entail that a is separable.

3.3. Matrices

Matrices represent linear operators on Hilbert spaces. We can add and multiply them, and use them to operate on vectors.

A typical example is $UVa = U(Va) = b$.

This means: apply the V transform to the vector a, then apply the U transform to the resulting vector, and the answer is the vector b.

One of the key properties of matrices is that they define linear operations, namely: $U(a + b) = Ua + Ub$.

All transformations are linear in quantum algorithms, so it is different from classical ones.

The linearity restriction both gives quantum algorithms great power and curiously makes them so different from classical ones.

The matrix I_N denotes the $N \times N$ identity matrix. If U is a matrix, then $U^k = UU \cdots U$ (k copies).

We use U^T as usual to denote the **transpose** of the matrix U. The inner product of two real vectors a and b is given by $< a, b >= \sum_{k=0}^{m} = a(k)b(k)$.

Unitary matrix: A **real** matrix U is **unitary** if $U^T U = I$.

Examples:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \text{ (Permutation matrix)}$$

$$H_2 = H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ (Basic Hadamard matrix)}$$

$$X = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \text{ (Symmetry+Permutation)}$$

Hadamard matrix:

$$H_4 = H \otimes H = \frac{1}{2}\begin{bmatrix} H & H \\ H & -H \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Another definition of a unitary matrix is based on the notion of **orthogonality**. Call two vectors a and b orthogonal if their inner product is 0.

A matrix U is unitary proved each row is a unit vector and any two distinct rows are orthogonal.

The reason these matrices are so important is that they **preserve the Euclidean length:** $||Ua|| = ||a||$.

3.4 Complex Spaces and Inner Products

Conjugation of complex: a^*

$\overline{x+iy} = x - iy$ where x, y are real numbers and $i = \sqrt{-1}$.

Conjugation of matrix: U^*, where $U^*(i,j) = (U(i,j)^*)$.

Transpose Conjugation of matrix: $U^\dagger$

where $U^\dagger(i,j) = U^*(j,i)$.

Adjoint of matrix:

We now define the adjoint of a matrix U to be the matrix $V = U^\dagger$ such that $V[r,c] = \overline{U[c,r]}$.

Unitary matrix:

A complex matrix U is unitary provided $U^\dagger U = I$.

(Note that U^* is $U^\dagger$ in this book)

Inner product: $< a, b >= \sum_{k=0}^{m} \overline{a(k)} b(k)$.

If U is $m \times n$ and V is $r \times s$, then $W = U \otimes V$ is the $mr \times ns$ matrix whose action on product vectors $c(ij) = a(i)b(j)$ is as follows: $(1 \leq i \leq n, 1 \leq j \leq s)$

$$(Wc)(ij) = (Ua)(i)(Vb)(j).$$

where a is a n-dimension vector, and b is a s-dimension vector, $c = a \otimes b$.

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \end{bmatrix} \qquad V = \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \\ v_{21} & v_{22} & v_{23} & v_{34} \end{bmatrix}$$

$$W = U \otimes V = \begin{bmatrix} u_{11}V & u_{12}V & u_{13}V \\ u_{21}V & u_{22}V & u_{23}V \end{bmatrix}$$

The action Wd is well defined via the same linear combination of the outputs on the basis vectors. That is, if

$$d = \sum_{i=1}^{n} \sum_{j=1}^{s} d_{i,j} e_i \otimes e_j$$

then

$$Wd = \sum_{i=1}^{n} \sum_{j=1}^{s} d_{i,j} (Ue_i) \otimes (Ve_j)$$

Note that it does not matter whether the scalars $d_{i,j}$ are regarded as multiplying e_i, e_j.

We mainly use tensor products to combine operations that work on separate halves of an overall index ij.

Basic and Important Properties:

Let U, V be unitary matrices, and a, b are two states in the Ball Sphere $\mathbb{S}$. We have the following useful properties:

(1) The Close Property:

(1.1) $Ua \in \mathbb{S}$, then $||Ua|| = ||a|| = 1$;

($|||Ua||^2 = (Ua)^\dagger(Ua) = a^\dagger(U^\dagger U)a = a^\dagger(I)a = a^\dagger a = ||a||^2$).

(1.2) $a \otimes b \in \mathbb{S}$;

(1.3) Both UV and $U \otimes V$ are unitary matrices.

(2) The coordinate relation:

$$(U \otimes V)(a \otimes b) = (Ua) \otimes (Vb), \text{ and}$$
$$(U \otimes V)(a \otimes b)(xy) = (Ua)(x)(Vb)(y)$$

where $order(U) = order(a) = 2^n, order(V) = order(b) = 2^m, x \in \{0,1\}^n, y \in \{0,1\}^m$.

Basic computations based on Dirac Notations:

Transpose conjugation of vector $|\psi\rangle$: $\langle\psi|$.

Transpose of matrix A: A^T

Complex conjugation of matrix A: A^*

Inner product of vectors $\langle\varphi|\psi\rangle$: $\langle\varphi|\psi\rangle$

Product of vectors (as matrices) $|\psi\rangle\langle\varphi|$ is a matrix.

For a basic vector e_k of N-dimension, $|k\rangle\langle k| = E$ is a matrix of N-order, and

$$E(i,j) = \begin{cases} 1 & if\ \ i = j = k \\ 0 & \quad o.w. \end{cases}$$

Inner product by transformation matrix A:

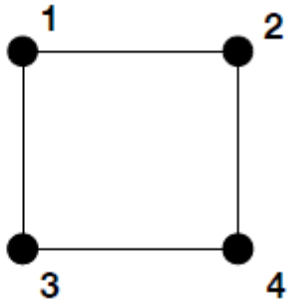
$$\langle\varphi|A|\psi\rangle =< \varphi, A\psi >.$$

Tensor product of vectors $|i\rangle \otimes |j\rangle$: $|i\rangle|j\rangle = |ij\rangle \qquad (e_x \otimes e_y = e_{xy})$;

3.5 Matrices, Graphs, and Sums Over Paths

A graph G consists of a set V of vertices together with a binary relation E on V . The adjacency matrix $A = A(G)$ of a graph $G = (V, E)$ If E is a symmetric relation, then A is a symmetric matrix. In that case, G is an **undirected graph**; otherwise it is **directed**.

A graph is **regular** of degree d if every vertex has degree d. In that case, consider the matrix $A' = \frac{1}{d}A$. For example, a four-cycle, C_4.



$$\boldsymbol{A}'_G = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \qquad \boldsymbol{U}_G = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix}.$$

A' is a stochastic (the rows sum to 1), and A' is doubly stochastic (the rows and columns sum to 1).

However, A' is not unitary for two reasons. First, the Euclidean norm of each row and column is $(\frac{1}{2})^2(1+1) = \frac{1}{2}$, not 1. Second, not all pairs of distinct rows or columns are orthogonal.

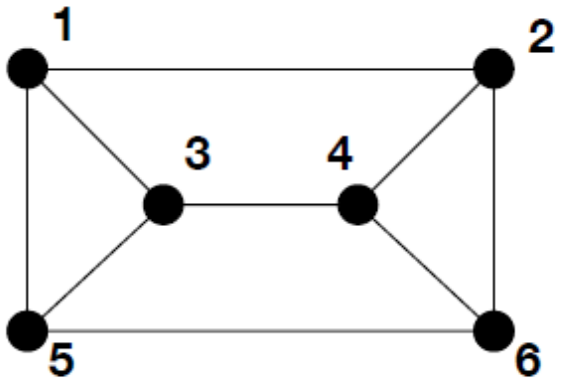
To meet the first criterion, we multiply by $\frac{1}{\sqrt{2}}$ instead of $\frac{1}{2}$. To meet the second, we can change the entries for the edge between nodes 3 and 4 from 1 to -1, creating the matrix U_G. Then U_G is unitary.

In fact, it is the tensor product $H \otimes X$ of two 2×2 unitary matrices.

$$\boldsymbol{A}'_G = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \qquad \boldsymbol{U}_G = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix}.$$

Some regular graphs can be converted into a unitary matrix.

But, an example of a d-regular graph whose adjacency matrix cannot similarly be converted into a unitary matrix.



$$\boldsymbol{A}' = \frac{1}{3}\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

We will see a general technique involving a tensor product of A_G with another matrix to create a **unitary matrix**, one representing a **quantum** rather than a classical **random walk** on the graph G.

The square of an adjacency matrix has entries given by

$$A^2[i,j] = \sum_k A[i,k]A[k,j].$$

The sum $A^2[i,j]$ counts the number of paths of length 2 from i to j. Likewise, $A^3[i,j]$ counts paths of length exactly 3, and so on.

For directed graphs in which each edge goes from one of n sources to one of m sinks, the adjacency matrices $A_{n,m}$ lend themselves to a path-counting composition. Consider

$$U = A_{n,m_1} A_{m_1,m_2} A_{m_2,m_3} \cdots A_{m_k,r}.$$

Then $U[i,j]$ counts the number of paths in the product graph that begin at source node i of G_1 and end at sink node j of the last graph G_{k+1}.

Key ideas:

When these are substituted for the "1" entries of the adjacency matrices, the value $U[i, j]$ becomes a sum of products over the possible paths.

In the classical case, these weights can be probabilities of individual choices along the paths.

In the quantum case, they can be complex **amplitudes**, and some of the products in the sum may **cancel**. Either way, the leading algorithmic structure is that of a **sum over paths**.

Some important basic unitary matrices:

Pauli matrices:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Twist matrix:

$$T_\alpha = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = T_\pi$$

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = T_{\frac{\pi}{2}} \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} = T_{\frac{\pi}{4}}$$

Rotation matrix:

$$R_\theta = \begin{bmatrix} cos(\theta/2) & sin(\theta/2) \\ -sin(\theta/2) & cos(\theta/2) \end{bmatrix}$$

Some results:

(1) For every 2×2 unitary matrix U, there are real numbers $\theta, \alpha, \beta, \delta$ such that

$$U = e^{i\delta} T_\alpha R_\theta T_\beta.$$

Thus, every 2×2 unitary operation can be decomposed into a rotation flanked by two twists, multiplied by an arbitrary phase shift by δ.

(2) The four Pauli matrices, I, X, Y, and Z, form an orthonormal basis for the space of 2×2 matrices, regarded as a 4-dimensional complex Hilbert space.

(3) $\frac{\pi}{8}$-Gate: The matrix V can be written as a composition of Hadamard H and T matrices.

$$V = \frac{1}{\sqrt{2}} \begin{bmatrix} e^{i\pi/4} & e^{-i\pi/4} \\ e^{-i\pi/4} & e^{i\pi/4} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix}$$

An important tools:

If both U and V are unitary matrices, then

(1) $U \otimes V$ is unitary too;

(2) for vectors a and b with proper dimensions, we have

$$(U \otimes V)(a \otimes b) = (Ua) \otimes (Vb).$$

Applications: (1) If $U_1, \cdots, U_k$ are 2×2 unitary matrices, and e_0 and e_1 are basic states, then

$$\begin{aligned}
&(U_1 \otimes U_2 \otimes \cdots \otimes U_k) e_{i_1 i_2 \cdots i_k} \\
&= (U_1 \otimes U_2 \otimes \cdots \otimes U_k)(e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_k}) \\
&= (U_1 e_{i_1}) \otimes (U_2 e_{i_2}) \otimes \cdots \otimes (U_k e_{i_k}).
\end{aligned}$$

where the string $i_1 i_2 \cdots i_k \in \{0,1\}^k$.

Applications: (2) Constructing unitary matrices:

Let U be a 2×2 unitary matrices, and I_2 be the identity matrix of order two. We can construct the following matrices:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes U \mapsto \begin{bmatrix} U & 0_{2\times 2} \\ 0_{2\times 2} & U \end{bmatrix} \mapsto \begin{bmatrix} I_2 & 0_{2\times 2} \\ 0_{2\times 2} & U \end{bmatrix}$$

Generally, we have:

$$\begin{bmatrix} I_{2^k} & \\ & U \end{bmatrix} \text{ where } U \text{ is an unitary matrix of order } 2^k.$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \mapsto \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mapsto \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$