Chapter 1

- **conventions**
  - n is a positive integer
  - $N = 2^n$ the number of binary strings of length n.
    (The number of n-bit binary strings).
  - When convenient we think of an n-bit binary string as a positive integer via its base 2 representation.
  - The k^{th} component of a vector $\mathbf{a}$ will be denoted by either a_k or $\mathbf{a}(k)$. This is the coefficient of $\mathbf{e}_k$ in the representation of $\mathbf{a}$ with respect to the standard basis: $\mathbf{a} = \sum_{k=0}^{N-1} a_k \mathbf{e}_k$.
- **state** of our (quantum) system is a vector in $\mathbb{R}^N$ (or $\mathbb{C}^N$) of length 1 (a *unit* vector).
  - Some examples in $\mathbb{R}^4$: $\begin{pmatrix} 2/5 & 4/5 & 2/5 & 1/5 \end{pmatrix}^T$ $\begin{pmatrix} 4/9 & 6/9 & -2/9 & 5/9 \end{pmatrix}^T$
  - An example in $\mathbb{C}^4$: $\begin{pmatrix} 1-i/4 & 1+2i/4 & 2-i/4 & -2i/4 \end{pmatrix}^T$
  - Another way to think of the computation of length:
    $\| \mathbf{a} \|^2 = \mathbf{a}^* \mathbf{a} = \overline{\mathbf{a}}^T \mathbf{a}$
- **transformations** of our state will be multiplication by matrices that preserve length. These matrices are called *unitary*.
  - An example of multiplication by a matrix $M = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$. Observe by example this doesn't preserve length so M is *not* unitary.
  - A way to see if U is unitary without checking $\| U\mathbf{a} \| = \| \mathbf{a} \|$ for all $\mathbf{a}$.
    Claim: A matrix U is unitary iff $U^* = U^{-1}$.
    * Recall(?) that, for matrices, $(UV)^T = V^T U^T$ and, for complex numbers, $\overline{wz} = \overline{w}\overline{z}$ and $\overline{w+z} = \overline{w} + \overline{z}$.
    * Conclude that, for matrices, $(UV)^* = V^* U^*$.
    * Compute $\| U\mathbf{a} \|^2 = \cdots = \| \mathbf{a} \|^2$.

    We've shown that matrices satisfying $U^* = U^{-1}$ are unitary. The other implication can be established with a little fiddling. We'll skip it.

---

start: with state vector equal to the basis vector $\mathbf{e}_0$.

move: repeatedly multiply state vector by unitary matrices.

Strangely, the input is encoded by the choice of unitary matrices.

end: *measure* final state $\mathbf{a}$ getting n-bit binary string k with probability $| \mathbf{a}(k) |^2$.

This is where the output is. Note that there is a nonzero probability that we'll get a k that is incorrect. In this case we run the whole thing again. The idea is to choose *moves* that stack the deck in favor of getting the correct answer. Shor's factorization algorithm is a good example of how this works.

Chapter 2

Asymptotic Notation

We restrict our attention to functions $f : \mathbb{N} \to \mathbb{N}$. In particular, we don't have to worry about dividing by 0. Just for fun let

$$\begin{aligned} f(n) &= 1^2 + 2^2 + \cdots + n^2 \\ &= (1/3)n^3 + (1/2)n^2 + (1/6)n \end{aligned}$$

Can prove this by induction.

Limits

$s(n) \sim t(n)$ if $\lim_{n\to\infty} s(n)/t(n) = 1$

We say $s(n)$ and $t(n)$ are asymptotically equivalent.

Ex. $f(n) \sim (1/3)n^3$

$s(n) = o(t(n))$ if $\lim_{n\to\infty} s(n)/t(n) = 0$

We say $s(n)$ is little-oh-of $t(n)$.

Ex. $f(n) = o(n^4)$

Bounds

$s(n) = O(t(n))$ if there is a positive constant c so that $s(n) \leq c \cdot t(n)$ for all n.

We say $s(n)$ is Big-Oh-of $t(n)$.

Note that this means that $s(n)/t(n)$ is *bounded away* from ∞.

Ex. $f(n) = O(n^4)$

Ex. $f(n) = O(n^3)$

Ex. $f(n) = (1/3)n^3 + O(n^2)$ *This needs interpretation.*

$s(n) = \Omega(t(n))$ if there is a positive constant c so that $s(n) \geq c \cdot t(n)$ for all n.

We say $s(n)$ is Big-Omega-of $t(n)$. This is the same as $t(n) = O(s(n))$.

Note that this means that $s(n)/t(n)$ is *bounded away* from 0.

Ex. $f(n) = \Omega(1)$

Ex. $f(n) = \Omega(n^3)$

$s(n) = \Theta(t(n))$ if $s(n) = O(t(n))$ and $s(n) = \Omega(t(n))$.

We say $s(n)$ is Big-Theta-of $t(n)$.

Note that this means that $s(n)/t(n)$ is *bounded away* from both 0 and ∞. This is not as strong as "$\lim_{n\to\infty} s(n)/t(n)$ exists and is some postive number."

Ex. $f(n) = \Theta(n^3)$

Simple operations

Use the interpretation of Big-Oh in terms of sets of functions to explain (some of) the following.

$$\begin{aligned} s(n) &= O(s(n)) \\ c \cdot O(s(n)) &= O(s(n)) \qquad \text{if } c \text{ is a positive constant} \\ O(s(n)) + O(s(n)) &= O(s(n)) \\ O(O(s(n))) &= O(s(n)) \\ O(s(n))O(t(n)) &= O(s(n)t(n)) \\ O(s(n)t(n)) &= s(n)O(t(n)) \end{aligned}$$

History

- O-notation is from P. Bachmann in 1894 (*Analytische Zahlentheorie*).
- o-notation is from E. Landau in 1909 (distribution of prime numbers).
- Ω and Θ notations are from Knuth (*The Art of Computer Programming*).

Chapter 3

Summary

- We start with an n-bit string as input.
- Our state is a unit vector in an $N = 2^n$ dimensional *Hilbert space*.
- Our transformations are $N \times N$ *unitary* matrices.
- We want an *feasible* algorithm. By this we mean that its running time should be $O(n^k)$ for some positive constant k. This is written, slightly inconsistently but conveniently, in the text as $n^{O(1)}$. (This is also called *polynomial time*.)
- To multiply a general $N \times N$ unitary matrix times a general vector in N-dimensional space takes time $O(N^2) = O(2^{2n}) \neq n^{O(1)}$. So we have to be careful about which unitary matrices we use in our algorithms. That's the next item on the agenda:
- Find unitary matrices that are useful but lead to a feasible algorithm.

3.1 Hilbert Spaces

Recall Some Notation:

(1) $U[r, c]$ is the entry in row r and column c of U.

(2) V is the *transpose* of U (written $V = U^T$) if $V[r, c] = U[c, r]$.

(3) V is the *adjoint* of U (written $V = U^*$) if $V[r, c] = \overline{U[c, r]}$. Other names for the adjoint are *Hamiltonian conjugate* and *conjugate transpose*.

(4) Hardly worth mentioning, but if U is real then $U^T = U^*$.

(5) A (square) matrix U is *unitary* provided $U^*U = I$.

Inner Product

An N-dimensional vector space over $\mathbb{R}$ (or $\mathbb{C}$) is a *Hilbert* space if it has an *inner product*. Our vector spaces are *column spaces* and our inner product will always be the following standard one.

(1) Definition: $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_k \overline{\mathbf{a}(k)} \mathbf{b}(k) = \mathbf{a}^* \mathbf{b}$

(2) Properties: $\langle \mathbf{b}, \mathbf{a} \rangle = \overline{\langle \mathbf{a}, \mathbf{b} \rangle}$, $\quad \langle \mathbf{a}, \mathbf{b} + \mathbf{c} \rangle = \langle \mathbf{a}, \mathbf{b} \rangle + \langle \mathbf{a}, \mathbf{c} \rangle$, $\quad \langle \mathbf{a}, \beta \mathbf{b} \rangle = \beta \langle \mathbf{a}, \mathbf{b} \rangle$

(3) Relation to length: $\langle \mathbf{a}, \mathbf{a} \rangle = \sum_k \overline{\mathbf{a}(k)} \mathbf{a}(k) = \sum_k |\, \mathbf{a}(k) \,|^2 = \|\, \mathbf{a} \,\|^2$

(4) We say that two vectors are *orthogonal* (perpendicular) if $\langle \mathbf{a}, \mathbf{b} \rangle = 0$.

(5) Multiplication by unitary matrices preserves the inner product: $\langle U\mathbf{a}, U\mathbf{b} \rangle = \cdots = \langle \mathbf{a}, \mathbf{b} \rangle$.

(6) For arbitrary matrices A and B, $(A^*B)[r, c] = \langle A[\,:, r], A[\,:, c] \rangle$, i.e., the entry in row r and column c of A^*B is the inner product of column r of A and column c of B. Hence, unitary matrices have *orthonormal* columns.

"We will use $\mathbb{H}_N$ to denote this space of dimension N." No they don't, at least not consistently (see first sentence of Section 3.2 for example).