15-459 Self-Study

Tyler Yang

Contents

1 Toggling Qubits (Lecture 1)

Video #001/100 Table of Contents

Q: What's a quantum computer?

A: something that can natively represent qubits.

Well then, what's a qubit? A **qubit** is a data type that has special properties, and takes on one of two possible values: $|0\rangle$ or $|1\rangle$.

Let's run through a very simple set of instructions. Simulated results on the right.

1. `new qubit A, B, C, D` — $ABCD = |0000\rangle$
2. `toggle A` — $ABCD = |1000\rangle$
3. `if A then toggle B` — $ABCD = |1100\rangle$
4. `if (A and B) then toggle C` — $ABCD = |1110\rangle$
5. `if A then toggle D` — $ABCD = |1111\rangle$
6. `if D then toggle A` — $ABCD = |0111\rangle$
7. `extract all` — print $ABCD = 0111$

1.1 Exercises

Say that you are allowed to use the following quantum instructions:

1. `toggle ___`
2. `if ___ then toggle ___`
3. `if (___ AND ___) then toggle ___`

where the blanks may be filled in by distinct qubit variable names (like `A, B, C`, etc.)

(a) Suppose you want to write a little subroutine that implements `if (A OR B) then toggle C`. Show how to do it using only the three allowed instructions. After this operation, `A` and `B` should be in their original states.

(b) Same question but for implementing `if (NOT A) then toggle B`. After this operation, `A` should be back to its original state.

(c) Same but for `IncrMod4 A, B`, which treats the two bits `A, B` as the binary encoding of a number from $\{0, 1, 2, 3\}$ (with `A` as the most significant bit and `B` the least), and acts by incrementing this number by 1, modulo 4.

(d) Same but for `IncrMod8 A, B, C`.

(e) Same but for `IncrMod3 A, B`, which treats the two bits $\mathbf{A}, \mathbf{B}$ as the binary encoding of a number from $\{0, 1, 2\}$ and adds 1 mod 3 – and additionally just leaves the "invalid" case of $\mathbf{A}, \mathbf{B} = |11\rangle$ as-is.

(f) Same but for `SWAP A, B` (which does what you think it does)

(g) Same but for `LeftCyclicShift A, B, C` which sets **A**'s new value to **B**'s old value, sets **B**'s new value to **C**'s old value, and sets **C**'s new value to **A**'s old value. You can call subroutines you've previously written, if you want.

(h) Same but for `RightCyclicShift A, B, C`

(i) Go into python iteractive mode and type:

```
(x, y) = (459, 314)
x ^= y
y ^= x
x ^= y
(x, y)
```

What does it print out? Explain how/why this happened.

1.2 Solutions

Part (e): IncrMod3 of these exercises stumped me for a while. It's because **this problem is impossible if we define the question as follows:** the paths diagram is $|00\rangle \to |01\rangle$, $|01\rangle \to |10\rangle$, $|10\rangle \to |11\rangle$, $|11\rangle \to |11\rangle$, and we only use 2 qubits. Video #21/100 says it's solvable, Video #22/100 says you can't allocate new qubits. However, the following proof shows why this first definition of the problem is unsolvable:

Proof. Three allowed instructions are NOT, CNOT, and CCNOT
$\implies$ all allowed instructions are reversible, hence their corresponding path matrices are invertible (Video #008/100).
Path for `IncrMod3`: $0 \to 1$, $1 \to 2$, $2 \to 3$, and $3 \to 3$ $\implies$ $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (not invertible, 3 pivots)
$\implies \nexists$ matrix product of path matrices for NOT, CNOT, CCNOT = matrix for IncrMod3.
$\implies$ Cannot make `IncrMod3` (cannot create a non-invertible matrix from multiplying a bunch of invertible matrices) □

Therefore, we need to introduce an auxiliary qubit.

1. `new qubit C`
2. `if (A AND B) then toggle C`
3. `if C then toggle B`
4. `if B then toggle A`
5. `toggle B`

However, **this problem is solvable** using the alternative paths diagram for this problem ($|00\rangle \to |01\rangle$, $|01\rangle \to |10\rangle$, $|10\rangle \to |00\rangle$, $|11\rangle \to |11\rangle$) is solvable, and this is the specification Professor O'Donnell had in mind (written on whiteboard in Video #039/100).

(a) `def if (A OR B) then toggle C`

1. `if A then toggle C`
2. `if B then toggle C`
3. `if (A AND B) then toggle C`

(b) `def if (NOT A) then toggle B`

1. `toggle A`
2. `if A then toggle B`
3. `toggle A`

(c) `def IncrMod4 A B`

1. `if B then toggle A`
2. `toggle B`

(d) `def IncrMod8 A B C`

1. `if (B AND C) then toggle A`
2. `if C then toggle B`
3. `toggle C`

(e) `def IncrMod3 A B`

1. `toggle B`
2. `if B then toggle A`
3. `toggle B`
4. `if A then toggle B`
5. `if B then toggle A`
6. `if A then toggle B`

(f) `def SWAP A B`

1. `if A then toggle B`
2. `if B then toggle A`
3. `if A then toggle B`

(g) `def LeftCyclicShift A B C`

1. `SWAP A B`
2. `SWAP B C`

(h) `def RightCyclicShift A B C`

1. `SWAP A C`
2. `SWAP B C`

(i) The result is $(x, y) = (314, 459)$. We can explain this since XOR and CNOT do same operation; thus the sequence of XORs is the same as SWAP.

2 MysteryToggles

Video #002/100 Table of Contents

Consider a 6-bit MysteryToggles. Toggling one bit is equally informational as toggling 2. Therefore, to find which `if ___ then toggle Ans` operations are in MysteryToggles, we need to ask 6 questions. However, we can leverage the properties of qubits (superposition and the "Hadamard All" operation) to solve MysteryToggles in only one question.

1. `toggle Ans`
2. `Hadamard all`
3. `MysteryToggles`
4. `Hadamard all`
5. `Extract all`

will return 1 when the qubit has an `if ___ then toggle Ans` operation in MysteryToggles and 0 when the qubit isn't involved in MysteryToggles. This is the quantum advantage over classical computing.

2.1 Exercises

For the MysteryToggles problem with $\mathbf{X_1}, \mathbf{X_2}, \mathbf{X_3}, \mathbf{X_4}, \mathbf{X_5}, \mathbf{X_6}$, give a convincing explanation for why any "TogglesDetective" using only "classical" instructions must ask at least 6 questions to learn the secret contents of MysteryToggles.

2.2 Solutions

We will show generally for n qubits, not just 6. Reframe the question: Let there be n qubits.

$$\Omega = \{ \text{ possible contents of MysteryToggles } \}$$

Each $\omega \in \Omega$ can be represented as a binary string of length n, where the ith entry in the binary string is 1 if MysteryToggles contains the instruction `if` $\mathbf{X_i}$ `then toggle Ans`, and 0 otherwise. There are 2^n binary strings of length n, hence $|\Omega| = 2^n$.

Let $\omega_\top$ be the true contents of MysteryToggles. We want to find $\omega_\top$ using a series of questions. What are questions? Introduce TogglesDetective, which is is some combination of classical instructions, then at least one run of MysteryToggles to see the outcome of `Ans`. We say that each run of MysteryToggles is one "question" asked. Before asking a question, some of the n qubits will be toggled as a result of the classical instructions.

$$S = \{ \text{ indices of toggled bits before asking a question } \}$$

Using this state S for the n qubits, asking a question will return `Ans == 1` or `Ans == 0`. Prove that we need to ask at least n questions to find out $\omega_\top$ for certain.

Proof. WTS that each question cuts the number of possible outcomes for the contents of MysteryToggles in half at best. Thus, to get from 2^n possible outcomes down to 1 outcome $\omega_\top$, we need to ask at least $\log_2(2^n) = n$ questions.

Here are our possible questions:

- $|S| = 0$. Then no bits are toggled, so asking a question will definitely return `Ans == 0`. This doesn't give us any information, so the number of possible outcomes doesn't change.
- $|S| = 1$. For example, TogglesDetective may toggle $\mathbf{X_i}$, run MysteryToggles, and record the result. If MysteryToggles returns `Ans=1`, we know the contents of MysteryToggles contains the instruction `if` $\mathbf{X_i}$ `then toggle Ans`. Otherwise, it doesn't. Then, we've cut our number of possible outcomes in half.
- $1 < |S| \leq n$. For example, TogglesDetective may toggle $\mathbf{X_i}, \mathbf{X_j}, \ldots$, run MysteryToggles, and record the result. The possible operations we are testing are in the set $Q = \{$`if` $\mathbf{X_i}$ `then toggle Ans`, `if` $\mathbf{X_j}$ `then toggle Ans`, $\ldots\}$
  - If MysteryToggles returns `Ans=1`, we know an odd number of the instructions in Q are included in the contents of MysteryToggles.
  - If MysteryToggles returns `Ans=0`, we know an even number of the instructions in Q are included in the contents of MysteryToggles.

  Since the number of subsets of Q that have odd cardinality are roughly equal to the number of subsets of Q that have even cardinality, we cut our number of possible outcomes in half.

Since asking a question cuts down the possible outcomes by at most one half, we need to ask at least n questions to cut down 2^n possibilities down to one $\omega_\top$. □

3 What you will dislike about this course

Video #003/100 Table of Contents

This course doesn't teach physics, engineering details about present day quantum computers, quantum programming languages and frameworks, etc. However, it is possible that you can build everything in any quantum computing implementation with the stuff used in the exercises for lecture 1 + Hadamard all.

If/when large-scale quantum computers are built:

- will ruin a lot of current-day cryptography by solving factoring efficiently. (Shor, 1993)
- will solve SAT in time, not 2^n but 1.4^n (actually sqrt 2 to the n) (Grover, 1994)
- that's abt it for solving classical problems more efficiently.

There's only really two quantum computing paradigms: quantum fourier transform/phase estimation, and other stuff!

We're pretty confident quantum computers won't solve NP problems efficiently.

Now why would you study quantum computing?

1. learn the true nature of what's computable
2. only quantum computers can solve quantum problems

   similar to randomized/probabilistic computing.

   primality testing might be a thing for probabilistic computing, but even better with quantum computing and Shor's algo
3. fun!!

3.1 Exercises

Exercise: record videos of yourself explaining solutions to the exercises from videos 1 and 2.

3.2 Solutions

4 What is a qubit? (Lecture 2)

Video #004/100 Table of Contents

A computer is any programmable device that natively supports the data type bit.

- Bits: 0 and 1
- Computer programming = logical manipulation of bits = instructions for physical manipulation of objects
- can use anything to compute - get an object, get one of its properties, and assign some of the basic states to 0 and some of the basic states to 1.
  - coin (side up, heads or tails)
  - switch (position, up or down)
  - abacus bead (touching beam? yes or no)
  - horizontal bar magnet (side north is on, left or right)
  - die (face, 1,2,3,4,5,6)

A quantum computer is any programmable device that natively supports the data type qubit

- Qubits: $|0\rangle$ and $|1\rangle$ and combinations/superpositions
- literally any physical object with two basic states (in principle) can model a qubit
  - theoretically possible for a coin to be some combination of heads and tails at the same time
  - **Law of Quantum Mechanics:** if an object can be in two basic states (e.g., $|0\rangle$ and $|1\rangle$), then it can also be in any "superposition state" – "x amplitude on $|0\rangle$", "y amplitude on $|1\rangle$" where $x, y \in \mathbb{R}$ such that $x^2 + y^2 = 1$.
  - generalizes to an analogous law: an object with more basic states also has superposition states where the sum of the squares of the weights for the linear combination is 1. e.g., an object with states $|0\rangle, |1\rangle, |2\rangle$ can have a superposition state $x|0\rangle + y|1\rangle + z|2\rangle$ where $x^2 + y^2 + z^2 = 1$ and $x, y, z \in \mathbb{R}$.
- the amplitudes can also be complex numbers, but they're not necessary to learn quantum computing.

4.1 Exercises

We're not going to mention complex numbers much in this course, but I know you are all tough cookies and can handle them if necessary. As a reminder, a complex number z is of the form $a + bi$, where a and b are real numbers, and i satisfies $i^2 = -1$. Complex numbers can be added, subtracted, multiplied, and divided just like real numbers. Also, complex numbers have an absolute value (or "magnitude"): namely, if $z = a + bi$, then $|z| = \sqrt{a^2 + b^2}$.

Full form of the Law of Quantum Mechanics: if an object has d "basic states" denoted $|1\rangle, |2\rangle, \ldots, |d\rangle$, then the object can also be in any "superposition state" of the form

$$z_1 \text{ amplitude on } |1\rangle, z_2 \text{ amplitude on } |2\rangle, \ldots, z_d \text{ amplitude on } |d\rangle$$

where $z_1 \ldots z_d$ are complex numbesr satisfying $|z_1|^2 + |z_2|^2 + \cdots + |z_d|^2 = 1$. (In class, we are pretty much only going to get involved with amplitudes that are real numbers, and d's which are powers of 2.)

Besides "i", another funny number we're going to be talking about a lot soon is "$\sqrt{\frac{1}{2}}$". (In case you didn't know, in math the convention is that if $x \geq 0$ is a real number, $\sqrt{x}$ denotes the nonnegative number whose square is x).

(a) Answer both of the following: What is the value of $\sqrt{\frac{1}{2}}$ to the nearest 1 decimal place? Also, suppose you go to a store advertising "all prices have been reduced 30%" and then at the bargain rack a sign says "take an additional 30% off the advertised price." If you buy a t-shirt from the bargain rack, then its price is *what* fraction of the additional price?

(b) Work out whether each of the following is a valid quantum state:

(i) $\sqrt{\frac{1}{2}}$ amplitude on $|0\rangle$, $\sqrt{\frac{1}{2}}i$ amplitude on $|1\rangle$

(ii) $0.8i$ amplitude on $|0\rangle$, 0.6 amplitude on $|1\rangle$

(iii) 0.4 amplitude on $|0\rangle$, 0.9 amplitude on $|1\rangle$

(iv) $\frac{1+i}{2}$ amplitude on $|0\rangle$, $\frac{1-i}{2}$ amplitude on $|1\rangle$

(v) 0.7 amplitude on $|00\rangle$, 0.1 amplitude on $|01\rangle$, -0.1 amplitude on $|10\rangle$, 0.7 amplitude on $|11\rangle$

(vi) $0.7 + 0.1i$ amplitude on $|0\rangle$, $-0.1 + 0.7i$ amplitude on $|1\rangle$

(vii) 0.1 amplitude on $|00\rangle$, 0.2 amplitude on $|01\rangle$, 0.3 amplitude on $|10\rangle$, 0.4 amplitude on $|11\rangle$

(viii) $\cos\theta$ amplitude on $|0\rangle$, $\sin\theta$ amplitude on $|1\rangle$ where θ is an angle.

4.2 Solutions

(a) $\sqrt{\frac{1}{2}} = \frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2} \approx 0.707 = 0.7$ The advertised price is 30% off the original price, and we then pay 30% off that. If the original price is x, the advertised price is $0.7x$, which makes the final price $0.7 * 0.7x = 0.49x$. It follows that the price we pay is $\frac{49}{100}$ of the original price.

(i) will work. Sum of the squares of the magnitudes is $\frac{1}{2} + \frac{1}{2} = 1$

(ii) will work. Sum of the squares of the magnitudes is $0.64 + 0.36 = 1$

(iii) won't work. $0.4^2 + 0.9^2 = 0.16 + 0.81 = 0.97$

(iv) will work. $|\frac{1+i}{2}| = |\frac{1+i}{2}| = \sqrt{(\frac{1}{2})^2 + (\frac{1}{2})^2} = \sqrt{\frac{1}{2}}$. similar magnitudes as part (i)

(v) will work. $|0.7|^2 + |0.1|^2 + |-0.1|^2 + |0.7|^2 = 0.49 + 0.01 + 0.01 + 0.49 = 1$

(vi) will work. $(\sqrt{0.7^2 + 0.1^2})^2 + (\sqrt{0.7^2 + (-0.1)^2})^2 = \sqrt{0.5}^2 + \sqrt{0.5}^2 = 1$

(vii) won't work. $0.1^2 + 0.2^2 + 0.3^2 + 0.4^2 = 0.01 + 0.04 + 0.09 + 0.16 = 0.30$

(viii) will work. $\cos^2\theta + \sin^2\theta = 1$

5 The Physics of Qubits

Video #005/100 Table of Contents

What objects **do** practically exhibit superposition? What macroscopic devices can measure the properties of these objects?

- a photon - property: linear polarization
  - basic states: horizontal $|0\rangle$ and vertical polarization $|1\rangle$
  - superposition states are like diagonal polarization
- sunglasses with polarizing lenses:
  - plastic that acts as a "vertical polarizing filter"
  - horizontally polarized photons get blocked = measurement outcome 0
  - vertically polarized photons pass through = measurement outcome 1
  - note that when a superposition state photon hits the lens, it could pass or go through with a certain probability (though this presumably doesn't change the state of the photon?)
- an electron - property: spin
  - basic states: spin down $|0\rangle$ or spin up $|1\rangle$
  - superposition states combine spin
  - what is spin? smth abt intrinsic angular momentum, but that's not really pertinent here.
- Stern-Gerlach device: tube of metal that fits on a table, isolate one electron, and send it down the tube. you have phosphorescent screens like 2 mm apart, then make an electromagnetic field around the tube. Electrons with down spin will hit the bottom phosphorescent screen, while up spin electrons hit the upper screen.
  - TLDR: spin manifests in real world phenomena.
  - lower screen glows = measurement outcome 0.
  - upper screen glows = measurement outcome 1.
- many possibilities for superposition states, but once you put a particle in a superposition state into a measuring device, the outcome is random.
- **Law of Quantum Measurement:** given a measurement device for the basic states $|0\rangle$ and $|1\rangle$, suppose you measure a qubit in the state x amplitude on $|0\rangle$ and y amplitude on $|1\rangle$, the probability you see measurement outcome 0 is x^2, and the probability you see measurement outcome 1 is y^2.

5.1 Exercises

Write a "Probabilistic Computer Simulator" in your favorite programming language. This will be like the "Quantum Computer Simulator" I demoed in Scratch, but for probabilistic computing rather than quantum computing. In particular, all of the variables will be of data type "bit" rather than "qubit".

For simplicity, rather than having variable allocation via commands like `new bit A`, let's assume that the first input to your program will be a nonnegative integer n, and this indicates that the program will use n bit-valued variables, which will just be numbered $1, 2, \ldots, n$ rather than named. (As usual, assume all bits are initialized to the value 0.) You should ensure that there's absolutely no trouble at all in having, say,

$n = 100,000$ (or larger). The point of this problem, which will become clear in its future Parts II and III, is to see that this task is easily doable for very large values of n (assuming your copmuter can actually generate random bits).

Regarding output, for simplicity, rather than requiring an explicit instruction like `extract all`, your program should just automatically print out all of the final bit values at the end. Thus after receiving n as input, the remaining input wil be some code consisting of a list of bit-manipulation instructions.

The bit-manipulation instructions supported will be the same ones we saw in our first Lecture 1 quantum code example (except that they will operate on bits, rather than qubits) ... plus the one special probabalistic instruction `RNG` that produces randomness. For your parsing simplicity, we will use the short instruction names used in quantum computing rather than my pseudocode versions, namely

- NOT i (`toggle bit i`)
- CNOT i, j (`if bit i then toggle bit j`)
- CCNOT i (`if (bit i AND bit j) then toggle bit k`)
- RNG i (set bit i to 0 or 1 with probability $\frac{1}{2}$ each)

In the above, i, j, k stand for *distinct* bit numbers between 1 and n. Your program may assume that the input code is properly formatted. Given the input n and subsequent code (say, from a text file, although you may use a reasonable alternative system if that's convenient for you) your program should use (pseudo) randomness, as built into your programming language of choice, to simulate one run of the code, printing out the final values of all the bits at the end.

5.2 Solutions (PCS1)

Completed on CMU AFS servers.

`Test Code` ($n = 5$)

- new qubit $A_1 \ldots A_5$
- toggle A_1
- toggle A_2
- if A_2 then toggle A_3
- if A_1 AND A_2 then toggle A_3
- Random A_5

6 Creating and Destroying Qubits

Video #006/100 Table of Contents

Two kinds of quantum instructions

1. (main) state-manipulation instructions. Toggle, ifThenToggle, Hadamard All
2. (special) creation and destruction/measurement. New, measure, etc.

`new qubit Apple` creates qubit Apple, initializes it to basic state 0.
`new qubit Banana, Cherry` creates two more qubits

measuring/destroying: "extract all" command ≡ putting qubits into a measuring device

- user sees a basic outcome
- output bit(s)
- result is random as per Law of Quantum Measurement
- note that extract all DESTROYS the qubits. like freeing/deallocating the variable

Q: do you always have to extract all? Can I just extract one of the qubits?

A: yes! but, the math gets a little more confusing.

6.1 Exercises

Suppose we have three qubits `A, B, C` and some quantum code has manipulated them into the state described on the left (below). Then we execute the code on the right. " 0.9 amplitude on $|000\rangle$,
-0.4 amplitude on $|001\rangle$,
0.1 amplitude on $|010\rangle$,
0.1 amplitude on $|100\rangle$,
0.1 amplitude on $|111\rangle$ "

1. `new qubit D`
2. `extract all`
3. `extract all`
4. `new qubit A, B`
5. `extract all`

Verify that the state on the left is indeed a valid quantum state, and then describe the output of the 2nd, 3rd, and 5th lines of code.

6.2 Solutions

$$0.9^2 + (-0.4)^2 + 0.1^2 + 0.1^2 + 0.1^2 = 0.81 + 0.16 + 0.01 + 0.01 + 0.01 = 1.00$$

so we clearly see that the state is a valid quantum state. Now, what happens with the code?

1. `new qubit D` Creates a new qubit D, initialize to 0.
2. `extract all` At this point, line 2 can return a few things. We have 5 possible outcomes:
   - $|0000\rangle$ has a $0.9^2 = 0.81$ probability
   - $|0010\rangle$ has a $(-0.4)^2 = 0.16$ probability
   - $|0100\rangle$ has a $0.1^2 = 0.01$ probability
   - $|1000\rangle$ has a $0.1^2 = 0.01$ probability
   - $|1110\rangle$ has a $0.1^2 = 0.01$ probability
3. `extract all` Once we've measured at line 2, is going to output whatever was output at line 2. This is because line 2 essentially collapses the qubits into a basic state, so extracting them is just going to return that same basic state.
4. `new qubit A B`
5. `extract all` is going to output whatever was output at line 2, except we flip the first two bits to 0 since A, B are initialized to zero in line 4.

7 Classical Reversible Instructions (Lecture 3)

Video #007/100 Table of Contents

1. `toggle A -> NOT A`
2. `if A then toggle B -> CNOT A B`

   `if A then toggle A` is not valid. uniqueness amongst qubits.
3. `if (A and B) then toggle C -> CCNOT A B C`
4. `if (A or B) then toggle C`
5. `if (not A) then toggle B`
6. `swap A B`

why are these "classical reversible instructions"?

- "classical":
  - basic state in, basic state out
  - they make sense even if their inputs are just bits, not qubits
  - not all classical bit ops are allowed (e.g., assignment isn't valid because it destroys the previous state). All qubit-manipulation instructions must be "reversible"
- "reversible"
  - there's a corresponding "undo" instruction
  - the undo of toggle is just toggle!
  - the undo of ifThenToggle is also just itself!
  - actually, all 6 instructions above ar etheir own undo, but that's a coincidence. for example, LeftCyclicShift's undo is RightCyclicShift

7.1 Exercises

Make a new version of your Probabilistic Computer Simulator. The input should be as before, but the output should be different. Instead of *simulating* the code using (pseudo) randomness and printing all the final bit values, your code should compute the true, exact probability that the final bit values of $000\cdots0$ (all n bits 0). This will be a rational number, like maybe $\frac{3}{8}$.

I expect you'll find it would be majorly difficult to tolerate $n = 100,000$. In fact, it is OK if your code only handles n up to 10.

7.2 Solutions (PCS2)

Completed on CMU AFS servers.

8 Path Diagrams/Matrices for Instructions

Video #008/100 Table of Contents

The analog of the truth table for quantum instructions.

Let's encode every basic state into a vector. Then our instructions can be represented as matrices that act on the basic state vectors.

Representing NOT: Let $|0\rangle = (1,0)$ and $|1\rangle = (0,1)$.

The (unraveled) path diagram for NOT is $|0\rangle \to |1\rangle$ and $|1\rangle \to |0\rangle$.

Thus, we want to find a linear transformation such that
$T(|0\rangle) = |1\rangle \implies T((1,0)) = (0,1)$, and
$T(|1\rangle) = |0\rangle \implies T((0,1)) = (1,0)$

But then we've just broken this down into finding the associated matrix A for T! We do this as follows:

$$A = \begin{bmatrix} \mathbb{P}\left(|0\rangle \to |0\rangle\right) & \mathbb{P}\left(|0\rangle \to |1\rangle\right) \\ \mathbb{P}\left(|1\rangle \to |0\rangle\right) & \mathbb{P}\left(|1\rangle \to |1\rangle\right) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and indeed, if we perform some matrix multiplication,

$$A\,|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0(1)+1(0) \\ 1(1)+0(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \text{and} \qquad A\,|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0(0)+1(1) \\ 1(0)+0(1) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Representing CNOT (controlled-not): We now have four basic states to consider.

Let $|00\rangle = (1,0,0,0)$, $|01\rangle = (0,1,0,0)$, $|10\rangle = (0,0,1,0)$, and $|11\rangle = (0,0,0,1)$.

The (unraveled) path diagram for CNOT is $|00\rangle \to |00\rangle$, $|01\rangle \to |01\rangle$, $|10\rangle \to |11\rangle$, and $|11\rangle \to |10\rangle$.

Then our corresponding matrix should be

$$A = \begin{bmatrix} \mathbb{P}\left(|00\rangle \to |00\rangle\right) & \mathbb{P}\left(|00\rangle \to |01\rangle\right) & \mathbb{P}\left(|00\rangle \to |10\rangle\right) & \mathbb{P}\left(|00\rangle \to |11\rangle\right) \\ \mathbb{P}\left(|01\rangle \to |00\rangle\right) & \mathbb{P}\left(|01\rangle \to |01\rangle\right) & \mathbb{P}\left(|01\rangle \to |10\rangle\right) & \mathbb{P}\left(|01\rangle \to |11\rangle\right) \\ \mathbb{P}\left(|10\rangle \to |00\rangle\right) & \mathbb{P}\left(|10\rangle \to |01\rangle\right) & \mathbb{P}\left(|10\rangle \to |10\rangle\right) & \mathbb{P}\left(|10\rangle \to |11\rangle\right) \\ \mathbb{P}\left(|11\rangle \to |00\rangle\right) & \mathbb{P}\left(|11\rangle \to |01\rangle\right) & \mathbb{P}\left(|11\rangle \to |10\rangle\right) & \mathbb{P}\left(|11\rangle \to |11\rangle\right) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and indeed, if we perform some matrix multiplication,

$$A\,|00\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |00\rangle \qquad \text{and} \qquad A\,|01\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = |01\rangle$$

$$A\,|10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle \qquad \text{and} \qquad A\,|11\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle$$

Representing CCNOT (controlled-CNOT): We now have eight basic states to consider. We will not bother to draw the 8×8 matrix, for that would be a waste of space. The idea is the same: A_{ij} (the entry in the ith row and jth column of A) is the probability that you end up in state j from state i. We can get

state i by converting from ket to binary to decimal; for example, $|110\rangle \to 110_2 = 6_{10}$. Interestingly, the path diagram for CCNOT is simple: it's the identity transformation for states 0 to 5, while states 6 and 7 swap ($|110\rangle \to |111\rangle$ and $|111\rangle \to |110\rangle$).

Representing SWAP: We claim that the matrix for SWAP is $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, as the only states that are changed by swap are $|01\rangle \to |10\rangle$ and vice versa. We can construct a path diagram from a path matrix, and vice versa! Note that all of the previous matrices we've done are permutations of the identity matrix, hence they are invertible. This is why these are
$\to$ *classical* (in that they flip bits, so they have basic states for input and output)
$\to$ and *reversible* (permutation matrices are invertible b/c nonzero determinant) instructions!

8.1 Exercises

Say we build a new classical reversible instruction as shown below:

`LeftCyclicShift(A,B,C):`

1. `swap A B`
2. `swap B C`

Draw the path diagram for this operation `LeftCyclicShift`. You should be able to see that it is *not* its own "undo"; nevertheless, it *is* reversible. Write a program for its reverse (undo) operation, and also draw the associated path diagram.

8.2 Solutions

We draw the (unraveled) path diagram as such, with the paths for the basic states enumerated on the side.

- $|000\rangle \to |000\rangle \quad 0 \to 0$
- $|001\rangle \to |010\rangle \quad 1 \to 2$
- $|010\rangle \to |100\rangle \quad 2 \to 4$
- $|011\rangle \to |110\rangle \quad 3 \to 6$
- $|100\rangle \to |001\rangle \quad 4 \to 1$
- $|101\rangle \to |011\rangle \quad 5 \to 3$
- $|110\rangle \to |101\rangle \quad 6 \to 5$
- $|111\rangle \to |111\rangle \quad 7 \to 7$

By enumerating the states in the path diagram, it becomes easy to construct the matrix. Namely, $A_{00} = A_{12} = A_{24} = A_{36} = A_{41} = A_{53} = A_{65} = A_{77} = 1$, and the rest of the entries are 0.

If you've been doing the work thus far, recall from Video 1 that we already wrote RightCyclicShift. `RightCyclicShift(A,B,C):`

1. `swap A C`
2. `swap B C`

We draw the (unraveled) path diagram as such, with the paths for the basic states enumerated on the side.

- $|000\rangle \to |000\rangle \quad 0 \to 0$
- $|001\rangle \to |100\rangle \quad 1 \to 4$
- $|010\rangle \to |001\rangle \quad 2 \to 1$
- $|011\rangle \to |101\rangle \quad 3 \to 5$
- $|100\rangle \to |010\rangle \quad 4 \to 2$
- $|101\rangle \to |110\rangle \quad 5 \to 6$
- $|110\rangle \to |011\rangle \quad 6 \to 3$
- $|111\rangle \to |111\rangle \quad 7 \to 7$

By enumerating the states in the path diagram, it becomes easy to construct the matrix. Namely, $A_{00} = A_{14} = A_{21} = A_{35} = A_{42} = A_{56} = A_{63} = A_{77} = 1$, and the rest of the entries are 0.

8.2.1 Linear Algebra Extension:

And we can observe that the matrix for both `LeftCyclicShift` and `RightCyclicShift` are invertible since they have a pivot in each column (thus full column rank square matrix). Thus both operations are reversible. And funnily enough, the matrix for `LeftCyclicShift` is the transpose of the matrix for `RightCyclicShift`, since all we had to do is just swap the states! It follows that the matrix for `LeftCyclicShift` is an orthogonal matrix whose transpose is its own inverse!

9 Superposition-Creating Instructions

Video #009/100 Table of Contents

- Don't have basic state in, basic state out property.
- There's really only one that you'll ever need: Hadamard.

Hadamard

- only operates on one qubit: `Hadamard A`
- Hadamard all works because it doesn't matter which order you Hadamard your qubits!
- Hadamard $|0\rangle$ yields $\sqrt{\frac{1}{2}}$ amplitude on $|0\rangle$ and $\sqrt{\frac{1}{2}}$ amplitude on $|1\rangle$
- Hadamard $|1\rangle$ yields $\sqrt{\frac{1}{2}}$ amplitude on $|0\rangle$ and $-\sqrt{\frac{1}{2}}$ amplitude on $|1\rangle$

Q: Why does the negation matter, if when we extract, the probabilities will work out to be the same? (when squaring the amplitudes to find the probabilities, the negative goes away).

A: Once we start combining instructions, the negation thing does end up mattering.

- Path diagram now has weights. $|0\rangle \to \sqrt{\frac{1}{2}}|0\rangle + \sqrt{\frac{1}{2}}|1\rangle$, and $|1\rangle \to \sqrt{\frac{1}{2}}|0\rangle - \sqrt{\frac{1}{2}}|1\rangle$. The matrix for Hadamard is the 2×2 matrix $\begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \\ \sqrt{\frac{1}{2}} & -\sqrt{\frac{1}{2}} \end{bmatrix}$.

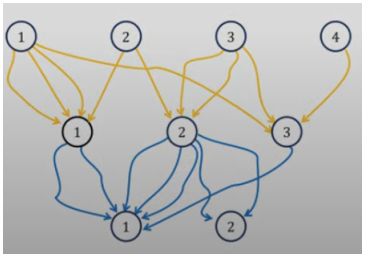
  Linear algebra side note: a Hadamard matrix is a square matrix with entries that are ± 1 and whose rows are mutually orthogonal. AKA, each pair of rows has matching entries in half the columns and mismatched entries in the remaining columns. It also follows that a Hadamard matrix's transpose is also a Hadamard matrix. Sylvester's construction recursively constructs Hadamard matrices: Given an $n \times n$ Hadamard matrix H, $\begin{bmatrix} H & H \\ H & -H \end{bmatrix}$ is a $2n \times 2n$ Hadamard matrix (Wikipedia). You'll note that the matrix for the Hadamard instruction is very similar to this Linear algebra definition, hence the name.
- Opcode `H`

if A then Hadamard B, AKA `CH` or "controlled-Hadamard"

- Controlled Hadamard has a similar path diagram, but we won't be using this much.

Clockwise A has path matrix $\begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix}$. **Counterclockwise A** has path matrix $\begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}$. These are rotation matrices (of the form $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$; Clockwise has theta where sin is negative, AKA theta is negative. Counterclockwise assumes theta is positive. The theta used is about 37.6°).

9.1 Exercises



Suppose you have a directed bipartite graph with m vertices on the top and n vertices on the bottom, where all the directed edges go from the top to the bottom. For clarity, call it the "amber"-colored graph, and let A be the $n \times m$ grid of numbers where in the ith column ($1 \leq i \leq m$) and jth row ($1 \leq j \leq n$) we have the number of ways (i.e., number of directed edges) from top vertex i to bottom vertex j. For example, in the above diagram (ignoring the blue lower half), we have $m = 4$, $n = 3$, and

$$A = \begin{bmatrix} 3 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Suppose we have another bipartite graph, the "blue" colored graph, with n vertices on the top, p vertices on the bottom, and associated $p \times n$ grid of numbers B. The lower half of the above diagram gives an example, with $p = 2$.

(a) Write B out for the specific blue graph shown in the diagram.

(b) Suppose we glue the n vertices at the bottom of the amber graph to the n vertices at the top of the blue graph, as in the diagram. Now let C be the $p \times m$ grid of numbers where in the ith column ($1 \leq i \leq m$) and kth row ($1 \leq k \leq p$) we have the number of ways (i.e., number of directed *paths*) from top vertex i all the way down to bottom vertex k. Write C out for the sepcific amber/blue combo graph shown in the diagram.

(c) In *general*, explain why C is derived from A and B from the rule for matrix multiplication, $C = BA$.

9.2 Solutions

(a) Let's go sequentially:

$1 \to 1$ 2 ways

$2 \to 1$ 3 ways

$3 \to 1$ 1 way

$1 \to 2$ 0 ways

$2 \to 2$ 2 ways

$3 \to 2$ 0 ways

B should be $p \times n = 2 \times 3$.

$$B = \begin{bmatrix} 2 & 3 & 1 \\ 0 & 2 & 0 \end{bmatrix}$$

(b) Let's go sequentially:

$1 \to X \to 1$ Case on intermediate X:

X is 1: $1 \to 1 \to 1$ has $3 \times 2 = 6$ paths

X is 2: $1 \to 2 \to 1$ has $0 \times 3 = 0$ paths

X is 3: $1 \to 3 \to 1$ has $1 \times 1 = 1$ paths

$1 \to X \to 2$ Case on intermediate X:

X is 1: $1 \to 1 \to 2$ has $3 \times 0 = 0$ paths

X is 2: $1 \to 2 \to 2$ has $0 \times 2 = 0$ paths

X is 3: $1 \to 3 \to 2$ has $1 \times 0 = 0$ paths

$2 \to X \to 1$ Case on intermediate X:

X is 1: $2 \to 1 \to 1$ has $1 \times 2 = 2$ paths

X is 2: $2 \to 2 \to 1$ has $1 \times 3 = 3$ paths

X is 3: $2 \to 3 \to 1$ has $0 \times 1 = 0$ paths

$2 \to X \to 2$ Case on intermediate X:

X is 1: $2 \to 1 \to 2$ has $1 \times 0 = 0$ paths

X is 2: $2 \to 2 \to 2$ has $1 \times 2 = 2$ paths

X is 3: $2 \to 3 \to 2$ has $0 \times 0 = 0$ paths

$3 \to X \to 1$ Case on intermediate X:

X is 1: $3 \to 1 \to 1$ has $0 \times 2 = 0$ paths

X is 2: $3 \to 2 \to 1$ has $2 \times 3 = 6$ paths

X is 3: $3 \to 3 \to 1$ has $1 \times 1 = 1$ paths

$3 \to X \to 2$ Case on intermediate X:

X is 1: $3 \to 1 \to 2$ has $0 \times 0 = 0$ paths

X is 2: $3 \to 2 \to 2$ has $2 \times 2 = 4$ paths

X is 3: $3 \to 3 \to 2$ has $1 \times 0 = 0$ paths

$4 \to X \to 1$ Case on intermediate X:

X is 1: $4 \to 1 \to 1$ has $0 \times 2 = 0$ paths

X is 2: $4 \to 2 \to 1$ has $0 \times 3 = 0$ paths

X is 3: $4 \to 3 \to 1$ has $1 \times 1 = 1$ paths

$4 \to X \to 2$ Case on intermediate X:

X is 1: $4 \to 1 \to 2$ has $0 \times 0 = 0$ paths

X is 2: $4 \to 2 \to 2$ has $0 \times 2 = 0$ paths

X is 3: $4 \to 3 \to 2$ has $1 \times 0 = 0$ paths

It follows that:

$1 \to 1$ $6 + 0 + 1 = 7$ ways

$2 \to 1$ $2 + 3 + 0 = 5$ ways

$3 \to 1$ $0 + 6 + 1 = 7$ ways

$4 \to 1$ $0 + 0 + 1 = 1$ way

$1 \to 2$ $0 + 0 + 0 = 0$ ways

$2 \to 2$ $0 + 2 + 0 = 2$ ways

$3 \to 2$ $0 + 4 + 0 = 4$ ways

$4 \to 2$ $0 + 0 + 0 = 0$ ways

C should be $p \times m = 2 \times 4$.

$$C = \begin{bmatrix} 7 & 5 & 7 & 1 \\ 0 & 2 & 4 & 0 \end{bmatrix}$$

(c) We will do the matrix multiplication first, then explain.

$$\begin{aligned} BA &= \begin{bmatrix} 2 & 3 & 1 \\ 0 & 2 & 0 \end{bmatrix} \begin{bmatrix} 3 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 2(3) + 3(0) + 1(1) & 2(1) + 3(1) + 1(0) & 2(0) + 3(2) + 1(1) & 2(0) + 3(0) + 1(1) \\ 0(3) + 2(0) + 0(1) & 0(1) + 2(1) + 0(0) & 0(0) + 2(2) + 0(1) & 0(0) + 2(0) + 0(1) \end{bmatrix} \\ &= \begin{bmatrix} 6 + 0 + 1 & 2 + 3 + 0 & 0 + 6 + 1 & 0 + 0 + 1 \\ 0 + 0 + 0 & 0 + 2 + 0 & 0 + 4 + 0 & 0 + 0 + 0 \end{bmatrix} \\ &= \begin{bmatrix} 7 & 5 & 7 & 1 \\ 0 & 2 & 4 & 0 \end{bmatrix} \end{aligned}$$

By thinking of matrix multiplication as a bunch of dot products, we realize that each entry is a dot product between two vectors: the number of ways to get from top vertex i to bottom vertex j, and the number of ways to get from top vertex j to bottom vertex k. But wait, the former is encoded in A, and the latter in B! Thus, we multiply the parts in A, by the parts in B, hence A on the right and B on the left since matrix multiplication starts from the right, so we get $C = BA$!

10 Probability Trees

Video #010/100 Table of Contents What happens when we apply our known instructions to superposition states?

Digression: **probability trees**. The laws and rules for analyzing quantum code are exactly the same, except "probability" is replaced with "amplitude". Probabilistic computation is regular deterministic code plus some probabilistic instruction.

For analogy, we'll only study probabilistic code where:

- all variables are of datatype bit
- all use classical reversible instructions, just on bits
- just use one RNG instruction, `noise b` toggles b with probability $\frac{1}{3}$ (made-up)

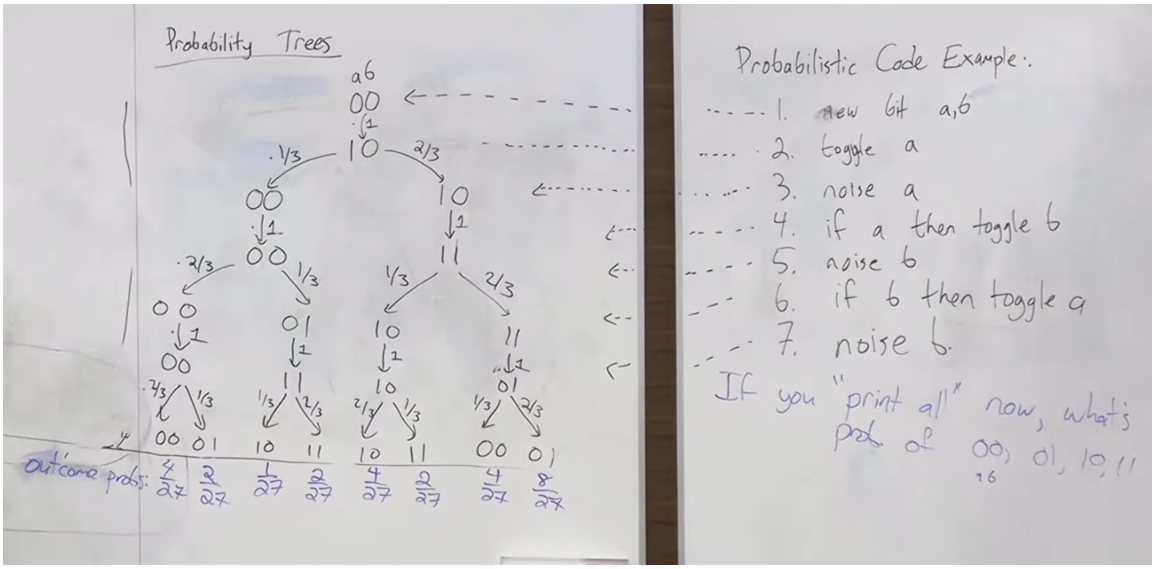
Path diagram for `noise b`:

- $0 \to 0$ has probability $\frac{2}{3}$
- $0 \to 1$ has probability $\frac{1}{3}$
- $1 \to 0$ has probability $\frac{1}{3}$
- $1 \to 1$ has probability $\frac{2}{3}$

Now let's write some probabilistic code:

1. `new bit a b`
2. `toggle a`
3. `noise a`
4. `if a then toggle b`
5. `noise b`
6. `if b then toggle a`
7. `noise b`

We'd draw a tree, easy to work out on a sheet of paper. However, there's a systematic way to do this analysis, which is tree analysis. It's a bit hard to LaTeX, so we'll just steal a screenshot from the video.



Now, the chance we get `ab = 00` by adding the probabilities of the branches that get that result. Same for all the possible results. This is also the motivation behind the exercise for Video 7 - though it took awhile beacuse I stupidly tried to code things in C.

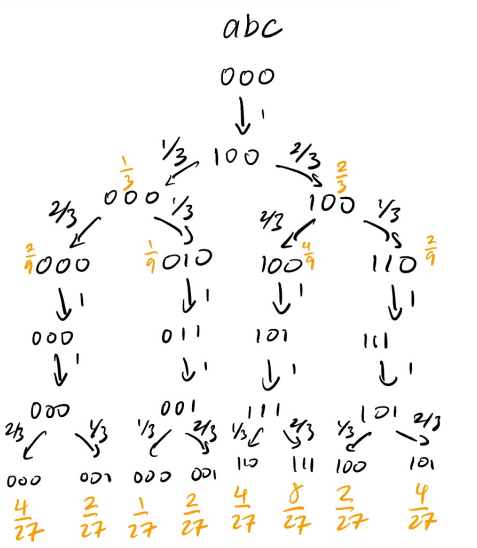
10.1 Exercises

Write down a probability tree for the code, and compute the probabilities of all 8 possible outputs.

1. `new bit a b c`
2. `toggle a`
3. `noise a`
4. `noise b`
5. `if (a or b) then toggle c`
6. `if c then toggle b`
7. `noise c`
8. `print all`

10.2 Solutions

For the sake of not having to LaTeXa tree, I will draw this in another software.



Verify: $4+2+1+2+4+8+2+4$

$= 7+6+8+6 = 15+12 = 27$ ✓

$P(000) = \frac{4+1}{27} = \frac{5}{27}$

$P(001) = \frac{2+2}{27} = \frac{4}{27}$

$P(010) = 0$

$P(011) = 0$

$P(100) = \frac{2}{27}$

$P(101) = \frac{4}{27}$

$P(110) = \frac{4}{27}$

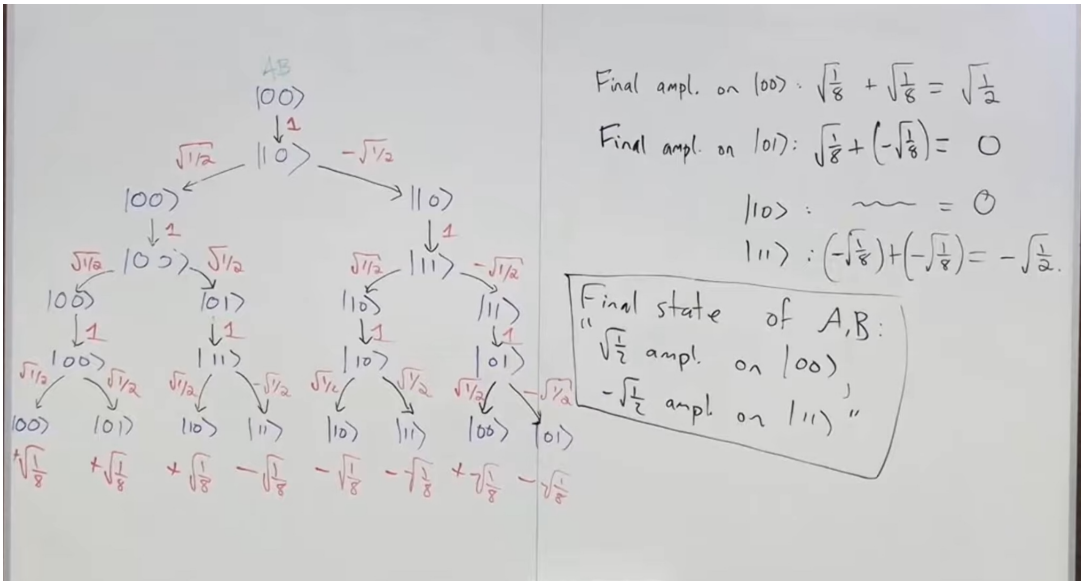
$P(111) = \frac{8}{27}$

11 Amplitude Trees (Lecture 4)

Video #011/100 Table of Contents Now let's do a quantum code example.

1. `new qubit A B`
2. `toggle A`
3. `Hadamard A`
4. `if B then toggle B`
5. `Hadamard B`
6. `if B then toggle A`
7. `Hadamard B`

If we "extract all" now, what's the probability of $|00\rangle, |01\rangle, |10\rangle, |11\rangle$? We can find this by drawing an amplitude tree.



There's no intuition behind this example. It's just an example. No think too hard.

Q: But we're getting two outputs – how does this lead to a deterministic answer?

A: Well, that's honestly a bit rare – we set things up so that we get the correct answer with high probability.

Q: How to interpret amplitude trees? They're not the same as probability trees in terms of meaning.

A: Umm just daydream about what it all really means. The math kind of represents physical experiments – it wasn't created from meaning, rather it was created to understand what we saw.

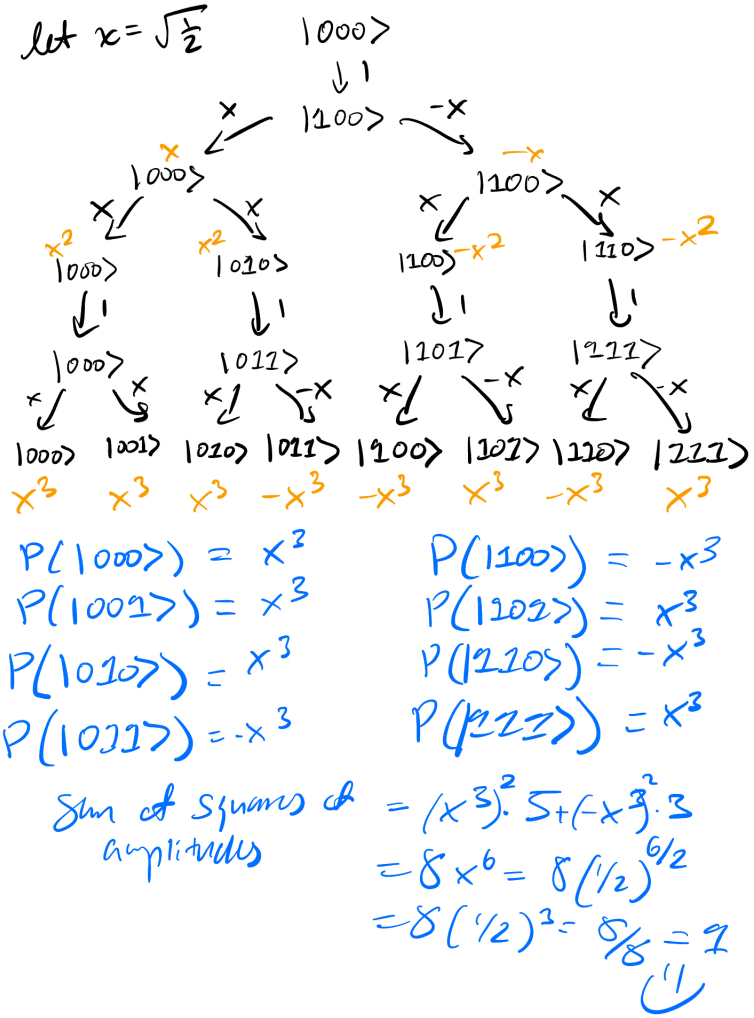
11.1 Exercises

Write down an amplitude tree for the code on the right, and compute the probabilities of all 8 possible outputs.

1. `new qubit A B C`

2. `toggle A`
3. `Hadamard A`
4. `Hadamard B`
5. `if (A or B) then toggle C`
6. `Hadamard C`
7. `extract all`

11.2 Solutions



12 Quantum Coin Flipping and Unflipping

Video #012/100 Table of Contents

Quantum Flip Coin

1. `new qubit A`
2. `Hadamard A`

Final state is $\sqrt{\frac{1}{2}}$ amplitude on $|0\rangle$ and $\sqrt{\frac{1}{2}}$ amplitude on $|1\rangle$. Extract all is $50-50$ for $|0\rangle$ and $|1\rangle$.

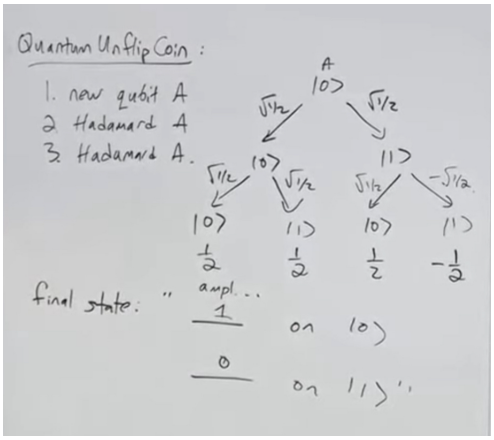
Quantum Flip Coin Part 2

1. `new qubit A`
2. `toggle A`
3. `Hadamard A`

Final state is $\sqrt{\frac{1}{2}}$ amplitude on $|0\rangle$ and $-\sqrt{\frac{1}{2}}$ amplitude on $|1\rangle$. Extract all is $50-50$ for $|0\rangle$ and $|1\rangle$.

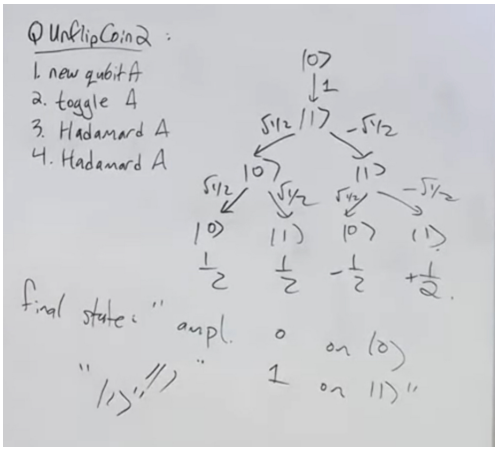
Quantum Unflip Coin

1. `new qubit A`
2. `Hadamard A`
3. `Hadamard A`



Quantum Unflip Coin Part 2

1. `new qubit A`
2. `toggle A`
3. `Hadamard A`
4. `Hadamard A`



What do we notice? Well, it appears that Hadamard undoes itself. However, let's think of this as flipping a coin. Let's say you flip a coin, heads 0 tails 1. Mathematically reasonable to say that the probability of 0 is one half and probability of 1 is one half. This quantifies your lack of knowledge of the true state of the coin.

However, you also realize that, actually, the true state of the coin is 0 or 1, not something in between. So Hadamard isn't really like coin flipping. The state of the coin flip is a superposition.

12.1 Exercises

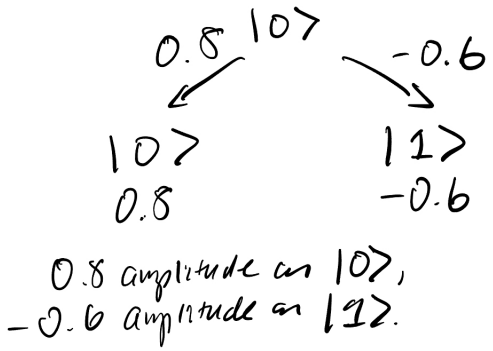
Work out the amplitude trees for the following two programs:

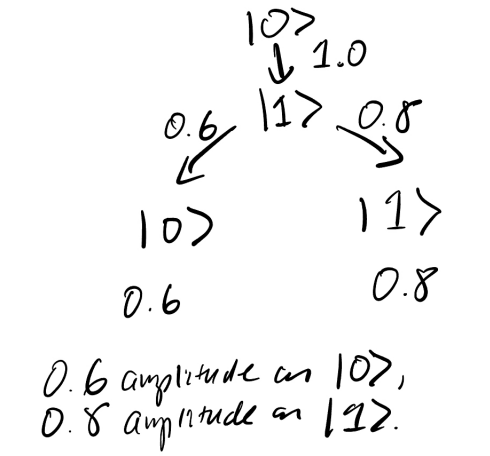
1. `new qubit A`
2. `clockwise A`
3. `counterclockwise A`

and

1. `new qubit A`
2. `toggle A`
3. `clockwise A`
4. `counterclockwise A`

12.2 Solutions





13 Composite Quantum Instructions

Video #013/100 Table of Contents

We can bundle elementary instructions into subroutines that become composite quantum instructions! Once you know how a subroutine operates on all the basic states, you know the behavior of the entire subroutine, and describe the composite instruction using a path diagram or matrix for the instruction! For example, as we learned from previous video, "double Hadamard" is a no-op. Hadamard is it's own undo.

Double Clockwise(A)

1. `Clockwise A`
2. `Clockwise A`

We can draw the path diagrams from the solutions to exercises from Video #12/100, but extend it one level down for each basic state. Consolidating subroutines into composite instructions has the effect of compressing the amplitude trees. Say you have two qubits A and B, and got them to the state "0.5 amplitude on $|00\rangle$, -0.5 amplitude on $|01\rangle$, -0.5 amplitude on $|10\rangle$, 0.5 amplitude on $|11\rangle$" To solve, you'd just draw the amplitude tree with our starting set of leaves.

13.1 Exercises

Let `MySub(B)` be a (qubit-manipulation) subroutine that operates on one qubit named `B`. (Everything in this problem works out just the same if `MySub` operates on more than 1 qubit, but let's just keep it simple.) Supose the matrix representation for `MySub` is $\begin{bmatrix} u & x \\ v & y \end{bmatrix}$.

(a) Consider the operation on two qubits `A,B` called `ControlledMySub(A,B)` which does the thing its name suggests: like, "`if A then MySub(B)`". Draw the paths matrix for `ControlledMySub(A,B)`.

(b) Consider the operation on two qubits `A,B` called `NoOpAndMySub(A,B)` which literally just does `MySub` on `B` and doesn't do anything to `A`. Draw the paths matrix for `NoOpAndMySub(A,B)`.

(c) Suppose there is another (qubit-manipulation) subroutine called `OtherSub(A)` operating on one qubit named `A` with the following matrix representation:

$$\begin{bmatrix} \alpha & \gamma \\ \beta & \varepsilon \end{bmatrix}$$

(still real numbers, just with Greek letters). Using an amplitude tree, work out the final state after this code:

```
1 new qubit A B
2 OtherSub A
3 MySub B
```

(d) Work out the final state after this code:

```
1 new qubit A B
2 MySub B
3 OtherSub A
```

(e) Explain why for *any* initial superposition state of two qubits, doing `OtherSub(A)` then `MySub(B)` yields the same final state as doing `MySub(B)` then `OtherSub(A)`.

13.2 Solutions

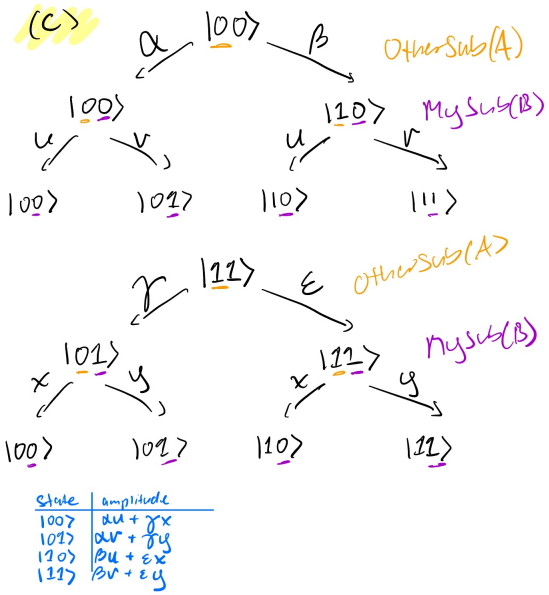
(a) $|00\rangle \to |00\rangle$ with amplitude 1
$|01\rangle \to |01\rangle$ with amplitude 1
$|10\rangle \to |10\rangle$ with amplitude u, $|10\rangle \to |11\rangle$ with amplitude v,
$|11\rangle \to |10\rangle$ with amplitude x, $|11\rangle \to |11\rangle$ with amplitude y.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u & v \\ 0 & 0 & x & y \end{bmatrix}$$

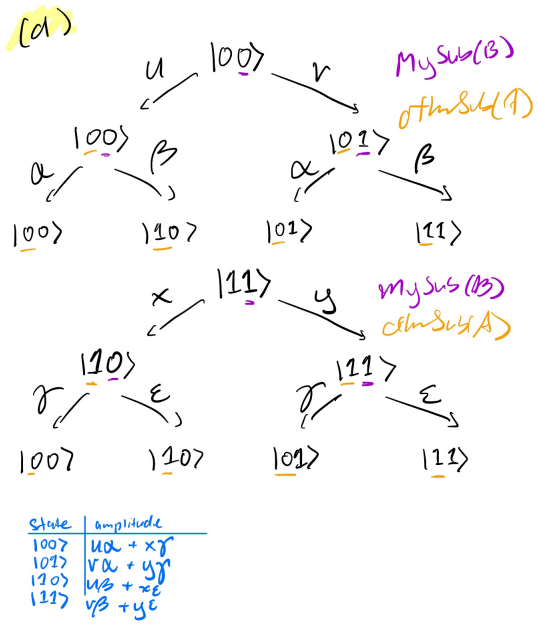
(b) $|00\rangle \to |00\rangle$ with amplitude u, $|00\rangle \to |01\rangle$ with amplitude v
$|01\rangle \to |00\rangle$ with amplitude x, $|01\rangle \to |01\rangle$ with amplitude y
$|10\rangle \to |10\rangle$ with amplitude u, $|10\rangle \to |11\rangle$ with amplitude v,
$|11\rangle \to |10\rangle$ with amplitude x, $|11\rangle \to |11\rangle$ with amplitude y.

$$\begin{bmatrix} u & v & 0 & 0 \\ x & y & 0 & 0 \\ 0 & 0 & u & v \\ 0 & 0 & x & y \end{bmatrix}$$

(c) See image below



(d) See image below



(e) `OtherSub(A)` and `MySub(B)` are independent operations. Neither operation relies on the state of the other bit to compute its result, so it would make sense that it doesn't matter what order you do them in.

Additional Observation: We can verify path matrices by computing their determinant. Since path matrices ensure that the before and after are both valid states, their determinant must have absolute value 1!

14 The Unitary Property

Video #014/100 Table of Contents

Q: What are all the possible operations we can get?

A: All quantum operations have the **Unitary Property:** given a *valid* state (squared amplitudes sum to 1), if you apply the op, the new state is also *valid*.

- **Law of Quantum Transformations:** Suppose a k-qubit manipulation operation has a $2^k \times 2^k$ matrix (paths diagram) with the Unitary Property. Then the transformation is physically possible (in principle). And conversely, these are all physically possible transformations.
- Let's check that the operations we already know are Unitary! If they are, then any subroutines we build with them (composite instructions) will also be Unitary!

  Theorem: every qubit manipulation Unitary operation can be programmed (technically, to arbitrarily good accuracy) using the 7 instructions (toggle, ifThenToggle, ifAndThenToggle, Hadamard, swap, etc.)

Q: Why are the classical reversible instructions Unitary?

A: Path diagrams look like an injective operation. Thus we can reverse them. If we think of classical reversible instructions as functions, they also have the same domain and codomain (the incoming state is a basic state, and the outcome is also a basic state, and all basic states are valid). You essentially just permute the amplitudes for the basic states.

14.1 Exercises

Here is the 4×4 matrix for a certain 2-qubit operation:

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

Show that this operation has the unitary property.

14.2 Solutions

For it to be unitary, given a valid state, we know the new state is valid. Let's apply the operation to an arbitrary valid state (a, b, c, d), where $a^2 + b^2 + c^2 + d^2 = 1$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a \\ -b \\ -c \\ -d \end{bmatrix}$$

We note that the final state is still valid, since $a^2 + (-b)^2 + (-c)^2 = (-d)^2 = a^2 + b^2 + c^2 + d^2 = 1$.

15 Hadamard is Unitary... (Lecture 5)

Video #015/100 Table of Contents

Q: Is Hadamard Unitary?

A: Let's perform Hadamard on an arbitrary valid state (x, y) where $x^2 + y^2 = 1$.

$$H(x,y) = \begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \\ \sqrt{\frac{1}{2}} & -\sqrt{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{1}{2}}x + \sqrt{\frac{1}{2}}y \\ \sqrt{\frac{1}{2}}x - \sqrt{\frac{1}{2}}y \end{bmatrix}$$

$$\left(\sqrt{\frac{1}{2}}(x+y)\right)^2 + \left(\sqrt{\frac{1}{2}}(x-y)\right)^2 = \frac{1}{2}(x^2+2xy+y^2)+\frac{1}{2}(x^2-2xy+y^2) = \frac{1}{2}(2x^2+2y^2) = x^2+y^2 = 1\checkmark$$

So Hadamard is Unitary.

15.1 Exercises

Verify 'clockwise' and 'counterclockwise' are unitary (and undoes of each other).

$$clockwise: \begin{bmatrix} .8 & -.6 \\ .6 & .8 \end{bmatrix} \qquad counterclockwise: \begin{bmatrix} .8 & .6 \\ -.6 & .8 \end{bmatrix}$$

15.2 Solutions

Fix arbitrary state (x, y) such that $x^2 + y^2 = 1$.

$$clockwise(x,y) = \begin{bmatrix} .8 & -.6 \\ .6 & .8 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} .8x - .6y \\ .6x + .8y \end{bmatrix}$$

$$\implies (.8x - .6y)^2 + (.6x + .8y)^2 = .64x^2 - .96xy + .36y^2 + .36x^2 + .96xy + .64y^2 = x^2 + y^2 = 1$$

$$counterclockwise(x,y) = \begin{bmatrix} .8 & .6 \\ -.6 & .8 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} .8x + .6y \\ -.6x + .8y \end{bmatrix}$$

$$\implies (.8x + .6y)^2 + (-.6x + .8y)^2 = .64x^2 + .96xy + .36y^2 + .36x^2 - .96xy + .64y^2 = x^2 + y^2 = 1$$

$$\begin{aligned} &counterclockwise(clockwise(x,y)) \\ &= counterclockwise(.8x - .6y, .6x + .8y) \\ &= (.8(.8x - .6y) + .6(.6x + .8y), -.6(.8x - .6y) + .8(.6x + .8y)) \\ &= (.64x - .48y + .36x + .48y, -.48x + .36y + .48x + .64y) \\ &= (x, y) \end{aligned}$$

$$\begin{aligned} &clockwise(counterclockwise(x,y)) \\ &= clockwise(.8x + 6y, -.6x + .8y) \\ &= (.8(.8x + 6y) - .6(-.6x + .8y), .6(.8x + 6y) + .8(-.6x + .8y)) \\ &= (.64x + .48y + .36x - .48y, .48x + .36y - .48x + .64y) \\ &= (x, y) \end{aligned}$$

16 Hadamard's Unitary Even With Other Qubits

Video #016/100 Table of Contents

- What if we Hadamard only on one qubit, but there are more qubits present?
- Say you have two qubits A, B in initial valid state (q, r, s, t) amplitudes on $|00\rangle$ $|01\rangle$ $|10\rangle$ $|11\rangle$
- Say we just Hadamard A. What is the new state?
- Pair them up into worlds cased on the value of B. Then it's like the one qubit case.
- "When you have multiple qubits, pair up the basic states baesd on the unchanging part. Then analyze the 1-qubit instruction on each pair."
- We'll do $\sqrt{\frac{1}{2}}(q+s)$ on $|0\rangle$ and $\sqrt{\frac{1}{2}}(q-s)$ on $|1\rangle$ when B is 0 and $\sqrt{\frac{1}{2}}(r+t)$ on $|0\rangle$ and $\sqrt{\frac{1}{2}}(r-t)$ on $|1\rangle$ when B is 1.

$\Longrightarrow$ Final state is $\sqrt{\frac{1}{2}}(q+s)$ on $|00\rangle$,
$\sqrt{\frac{1}{2}}(q-s)$ on $|10\rangle$,
$\sqrt{\frac{1}{2}}(r+t)$ on $|01\rangle$,
$\sqrt{\frac{1}{2}}(r-t)$ on $|11\rangle$.
Then final state is also valid, since the sum of squares becomes $\frac{(q+s)^2+(q-s)^2+(r+t)^2+(r-t)^2}{2}$
$= \frac{2q^2+2s^2+2r^2+2t^2}{2} = q^2+s^2+r^2+t^2 = q^2+r^2+s^2+t^2 = 1$.

16.1 Exercises

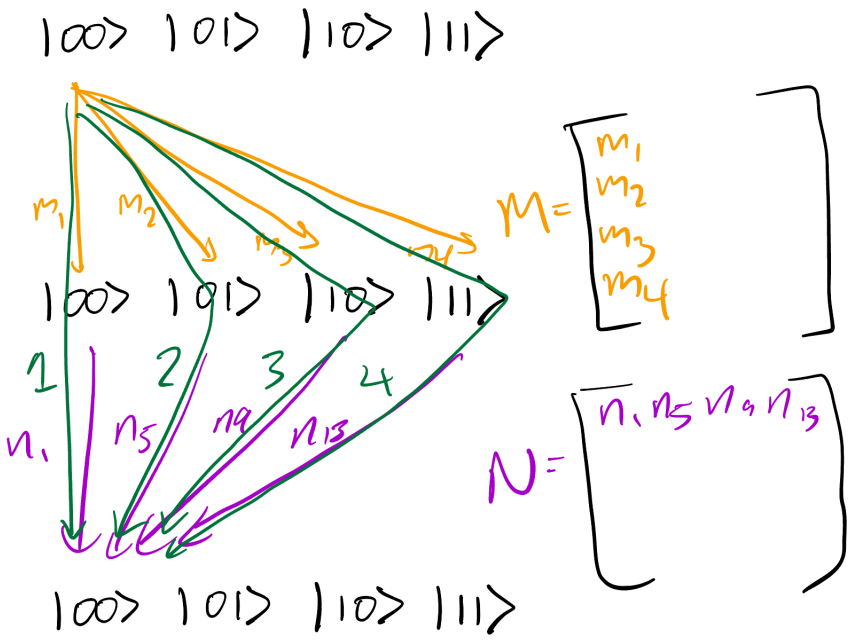
Suppose M is the 4×4 matrix representation of a quantum operation on 2 qubits. Given a matrix M, you're probabilty used to calling the entries of the matrix M_{ij}. But since we think of the columns/rows of M as being labeled by $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$, we will instead write $M[ab \to cd]$ for the entry in column $|ab\rangle$ and row$|cd\rangle$.

Next, suppose N is the 4×4 matrix representation of another operation on 2 qubits; we again use notation like $N[ab \to cd]$ for its entries.

Now suppose we consider the composite oepration on 2 qubits in which we first do (the operation represented by) M and then we do (the operation represented by) N. Show that the matrix representation of this complete operation is $N \cdot M$, the matrix product.

Note: Please remember that the only method you have at your disposal to analyze how quantum operations work is amplitude trees. Hint: consider gluging the paths diagram for M to the paths diagram for N... Also, I suggest you try doing an example with numbers first, as practice.

16.2 Solutions



17 Unnormalized States

Video #017/100 Table of Contents

- getting rid of $\sqrt{\frac{1}{2}}$ – don't write the $\sqrt{\frac{1}{2}}$'s
- For mathematical convenience, we'll multiply all amplitudes by some constant c. Then the sum of squares of amplitudes will be c^2.
- "unnormalized states"
- Multiply ALL amplitudes by the same amount.
- "Add & Diff A" is just Hadamard A but we just return output states $x+y$ on $|00\rangle$, $x-y$ on $|01\rangle$. This operation unnormalizes our state but it's basically Hadamard (physically the same).

17.1 Exercises

Suppose you have two paths diagrams called X and Y, where X has m vertices on top and n vertices on the bottom, and Y has n vertices on top and p on the bottom (and all arrows labeled by real numbers). Hopefully, you have come to understand the following:

- X is the same as an $n \times m$ matrix
- Y is the same as a $p \times n$ matrix
- If you glue together the n top nodes of Y to the n bottom nodes of X and then you "compress" the result to a single paths diagram Z (where the arrow from s to t in Z is labeled by the sum of products-along-paths-from-s-to-t in the glued diagram), then Z is equivalent to the matrix $Y \cdot X$.

(a) Explain why a paths diagram with a single top node and 2^k bottom nodes (labeled by k-bit strings) is equivalent to a column vector, and why it can also be interpreted as a k-qubit quantum state.

(b) Given an arbitrary k-bit superposition state ϕ, and a k-qubit manipulation operation Q, explain how to answer the question "What quantum state results from starting with state ϕ and applying Q?" using paths diagrams.

(c) Given any paths diagram X with m top vertices and n bottom vertices (aka $n \times m$ matrix), let $X^\dagger$ (pronounced "X-dagger") be the paths diagram/matrix formed by reversing all the arrows in X (in case X's entries have complex numbers, you should also do "complex-conjugate" on each of them when forming $X^\dagger$. Don't worry about this, because we're not going to use complex numbers, but I wanted to let you know).

(d) Suppose ϕ and ψ are both paths diagrams with one vertex on top and m vertices on bottom. In matrix terminology, what type of object is $\psi^\dagger$?

(e) Following up on the previous equation, what well-kown linear algebra concept arises when you glue $\psi^\dagger$ to the bottom of ϕ and compress the result down to a 1×1 path diagram?

(f) Following up some more, supposing $m = 2^k$, how could you use these ideas to know if ϕ stands for a valid k-qubit quantum state?

17.2 Solutions

(a) I've already been interpreting this as such in this document, because we can think of it this way: The 0th entry corresponds to $|00\rangle$, 1st entry to $|01\rangle$, and basically the ith entry corresponds to the amplitude for $|$the base 2 representation of $i\rangle$, where $0 \leq i \leq 2^k - 1$ (basically the base 2 representation of i goes from $0 \ldots 0$ to $1 \ldots 1$).
It's also a column vector because you only have one possible state in and a bunch of fixed output states whose amplitudes are given in the column vector.

(b) Find path matrix for Q. Find column vector for ϕ. Compute $Q\phi$.

(c) No answer to write here, but the dagger notation is from the Hermitian transpose (or conjugate transpose). $A^\dagger = A^H = (\bar{A})^T = \bar{A^T}$. Wikipedia.

(d) $\psi^\dagger$ is a row vector (you've transposed a column vector ψ, so now you get an $m \times 1$ matrix, AKA length m row vector)

(e) It's a dot product between the conjugates of ψ and the elements of ϕ. If ψ and ϕ both contain only real entries, it's just the dot product of the vectors ψ and ϕ.

(f) ϕ is a valid k-qubit quantum state if and only if the sum of squares of its entries is 1. But the sum of squares the entries of ϕ is equal to $\|\phi\|^2 = \phi \cdot \phi = \phi^T \phi$. For ϕ with complex entries, the norm is actually given by the dot product of ϕ with its complex conjugate $\bar{\phi}$, leading to

$$\phi^\dagger \phi = 1 \iff \phi \text{ is a valid } k\text{-qubit quantum state.}$$

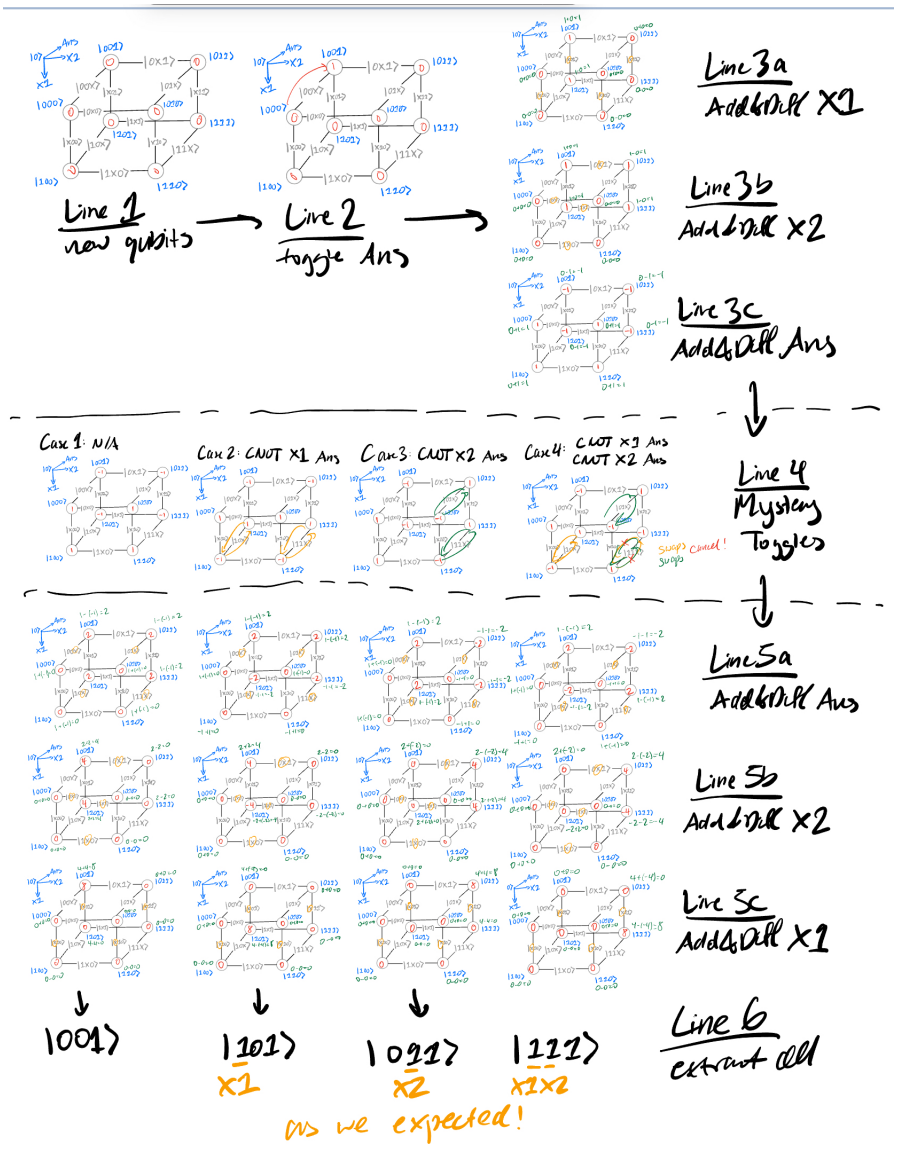
18 How TogglesDetective Works (Part 1)

Video #018/100 Table of Contents

MysteryToggles (just X_1, X_2, Ans) can be solved using the following TogglesDetective:

1. `new qubit` X_1, X_2, Ans
2. `toggle Ans`
3. `Hadamard all` Separate into `Add&Diff A Add&Diff B Add&Diff Ans`
4. `MysteryToggles`
5. `Hadamard all` Separate into `Add&Diff Ans Add&Diff B Add&Diff A`
   recall order doesn't matter when you operate on different qubits, exercises from Video #13/100
6. `extract all`

To analyze this, we can arrange them in a 3-D cube (3 qubits so 3 dimensions).



18.1 Exercises

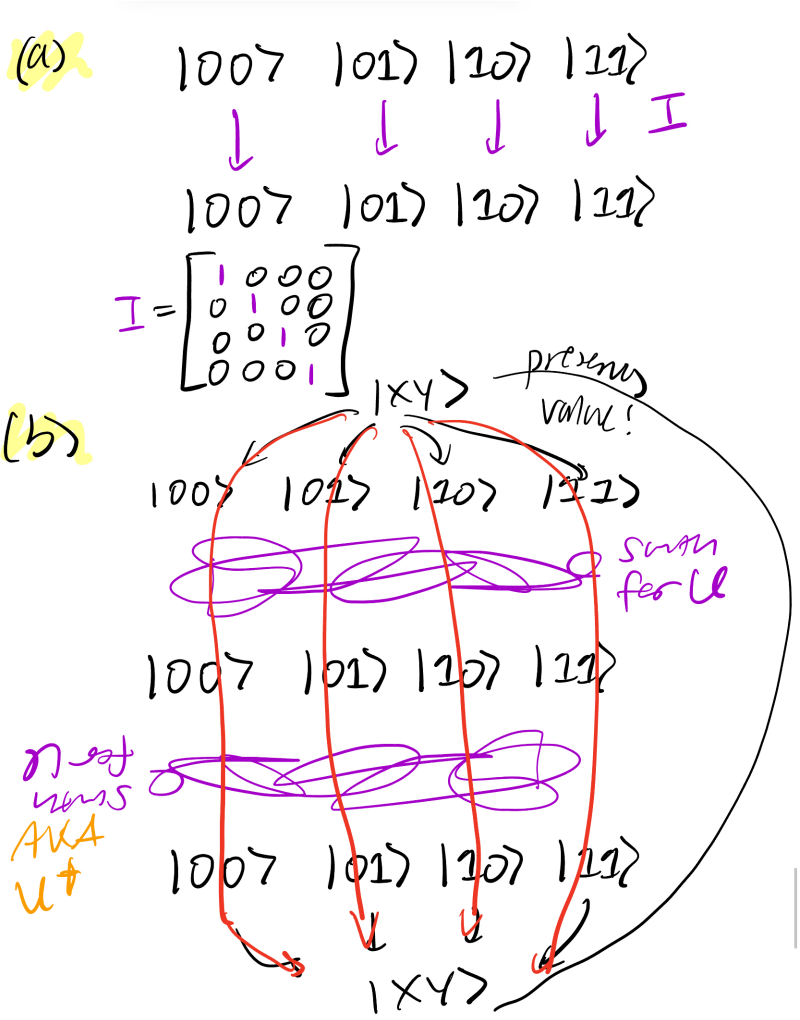
(a) Let I be 2-qubit-manipulation operation that literally does nothing to the two qubits. Draw the paths diagram and matrix for I.

(b) Suppose U is a 4×4 real matrix (this problem works just the same for any $2^k \times 2^k$ matrix, but let's stick to $k = 2$ just for concreteness). We think of U's columns/raws as being labeled by 2-bit strings, even though U might not correspond to a valid quantum transformation. But suppose that $U^\dagger \cdot U = I$. Show that U has the unitary property.

I require your proof to involve illustrative pictures in which a paths diagram with one top node and 4 bottom nodes is glued to a 4×4 paths diagram, and then this whole picture is also turned upside down.

18.2 Solutions

Could formalize my answer for exercise (b) by drawing 16 traces, each with labeled letters, or showing a subsection like I did in the exercises for Video #16/100. Bit tedious though.



19 How TogglesDetective Works (Part 2)

Video #019/100 Table of Contents See notes for the previous video.

19.1 Exercises

Work out the remaining two cases (Draw all the intermediate amplitude-labeled cube diagrams!):
Case 3: MysteryToggles is just "if X_2 then toggle Ans"
Case 4: MysteryToggles is "if X_1 then toggle Ans" and "if X_2 then toggle Ans"

19.2 Solutions

Done ahead of time, in notes for Video #18/100

Computing in Superposition & Hadamard (Fourier) Transform

20 Quantum Programming Paradigm 1 (Lecture 6)

Video #020/100 Table of Contents

Many of the following videos are going to deal with this first paradigm.

Suppose CCode (classical code) is a classical computer program that takes in n bits and outputs m bits.

1. Convert CCode to an "equivalent" quantum program QCode on n qubits

   MysteryToggles used only classical reversible instructions, so it is kinda its own QCode translation.

2. Prepare the "uniform superposition" (amplitude 1 on all 2^n basic states)
   note this is unnormalized state

   Hadamard all gets to the uniform superposition (or 3 Add&Diff instructions in our example)

3. Run QCode ... get all possible 2^n input/output pairs, somehow encoded in the amplitudes.

4. Somehow (?) get negative amplitudes involved

   this is why we toggled Ans in step 2

5. Hadamard all (kinda does a Fourier Transform? Much amplitude on important signals, little amplitude on nonimportant signals? Vague but this is a high-level summary)

   in step 5, we Hadamard all! It kinda collapsed everything together!

6. Extract all (profit?)

20.1 Exercises

This continues exercises from Video #5/100 and Video #7/100. Change your Probabilistic Computer Simulator so that it becomes a Quantum Computer Simulator. The input format should be *almost* the same as before, but instead of supporting the "RNG" instruction, you should support the "HAD" (=Hadamard) instruction (and of course, the variables now represent qubits, rather than bits). As for the ooutput, two things will be required.

- Very similar to Part II, your code should first output the exact final amplitude on $|00\ldots0\rangle$. As long as you output an easy-to-interpret representation of a number that exactly equals the final amplitude, I don't mind the precise format; I suppose best would be something like "`-(3/8)sqrt(2)`" or "`5/1024`".
- Second, somewhat similar to Part I, your code should additionally *simulate* (using the pseudorandom number generator built into the programming language you're using) what would happen if the code ended with an "extract all" instruction. (So the output should look like one amplitude followed by one n-bit string; if you rerun the code with the same input, the amplitude should always be the same, but the n-bit string will probably change.)

Did you find Part II of this problem harder to easier to code than Part I? Why?

What about versus this Quantum Part III? Should you have supported $n = 1000$ in Parts II or III?

Given a probablistic program P, suppose you wanted to get a decent sense for the fraction of times it outputs each possible n-bit string. And suppose you'd be happy to have output like in Part I, but the language you're

coding in has absolutely no support for (pseudo) randomness. Would you then be obliged to write your Part II code?

Conversely, suppose that given a quantum program Q, you don't care about the final amplitudes per se, you just want to get a decent sense for the fraction of times "extract all" would output each possible n-bit string. Are you obliged to write your Part III code? Might there be some kind of "simulation shortcut" as in Part I?

What if your favorite programming language natively supported actual qubits?

20.2 Solutions (QVM)

Completed on CMU AFS servers. Did not implement fraction reduction; instead, just alternated Add&Diff and Avg&Dev operations for Hadamards. Supporting $n = 1000$ in parts II or III would've required a states array of size 2^n; but $2^{100} = 1.2676506002 \times 10^{30}$, which is a bit too much memory for my poor laptop.

Part II was harder than Part I, but that's because I was being stupid at first. I was going to do a tree-based state implementation, where I hash state distributions into a table, but that was way too convoluted. I did have to write an entire fraction library in C, but it ended up being pretty good.

Quantum Part III and Part II were essentially the same; just replacing sum of probabilities with sum of squares of amplitudes, and replacing probabilty with Add&Diff or Avg&Dev for Hadamard. I actually went back to the exercises for Video #005/100, Video #007/100, and Video #020/100 after I had completed Video #042/100. The later videos didn't help in termrs of implementation, but I completed the exercises much quicker after understanding more of the course. Specifically, it took in total:

- 2 hours of setup for the structs and ideas, though it failed at first (right after finishing videos 5 and 7)
- 6 hours of failing to figure out how to do the tree-hashtable implementation I initially thought of
- a week later, took about 30 minutes of tinkering plus 5 hours of focused work later that evening to complete all three parts of the exercise (PCS1, PCS2, and QVM). Code is on private GitHub repo.

21 CCode to QCode: If f then Minus spec

Video #021/100 Table of Contents

- CCode computers a boolean function $f : \{0,1\}^n \to \{0,1\}^m$
- let $n = m = 2$ (bitstrings represent numbers in mod 4)

  $f =$"increment mod 4"(AKA, $f(x) = x + 1 \bmod 4$)
- let $n = 16, m - 8$. Input bitstrings represent two 8-bit numbers, output is their sum mod 2^8

  $f(x, y) = x + y$ (with overflow handled)

But what does it mean to convert CCode to equivalent QCode? Most of these things aren't even reversible, so you can't just use NOT, CNOT, and CCNOT.

What should QCode do?

- Special Case 1: $n = m$ and f happens to be reversible.

  we hope that QCode literally does f, just with qubits.
- Special Case 2: $m = 1$.

  literally impossible for quantum code to do this, because these operations are irreversible.

A candidate "spec" for QCode to check palindromeness.

"if $f(B_1, B_2, \ldots, B_n)$ then minus", defined as follows.

If the input basic state is $|b_1 b_2 \cdots{}_n\rangle$ and the output state should be 0, then we just output the input basic state. Else if the output state should be 1, we put amplitude -1 on the input basic state. If we think of the paths diagram/matrix, it ends up being the identity matrix, except some of the 1s have been swapped for -1s. So such an instruction is unitary, though not a classical reversible instruction.

Then, Hadamard all does something interesting on the state after the minus operation!

21.1 Exercises

(a) Consider the following quantum code:

```
Zee(B):
    Hadamard B
    toggle B
    Hadamard B
```

Determine, with justification, the paths diagram (or matrix, your choice) representing Zee. Also state the paths diagram (or matrix) you should get if the "toggle B" instruction were deleted.

(b) Consider the following quantum code:

```
SeeZee(A,B):
    Hadamard B
    If A then toggle B
    Hadamard B
```

This code is actually equivalent to "if $f(A, B)$ then minus" for a certain Boolean function $f : \{0,1\}^2 \to \{0,1\}$. Determine, with justification, which f this is.

21.2 Solutions

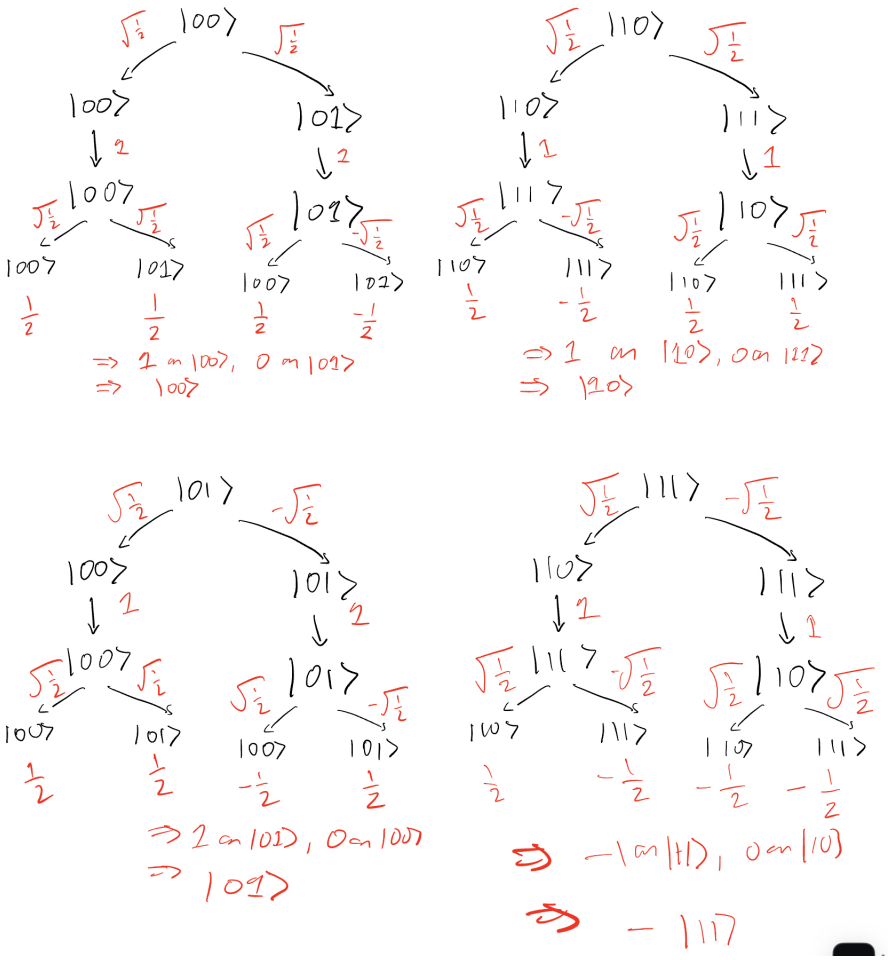
(a) The path matrix for Hadamard is $\begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \\ \sqrt{\frac{1}{2}} & -\sqrt{\frac{1}{2}} \end{bmatrix}$ by definition of Hadamard. The path matrix for toggle is $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ (we just swap the state). We can find the paths matrix for Zee through matrix multiplication:

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

If we remove the toggle B instruction,

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

(b) We do this analysis using a paths diagram.



Evidently, this is the function $f(a, b) =$ "a equals b" and $a, b = 1$.

22 CCode to QCode, the general spec

Video #022/100 Table of Contents

The stuff from Video 21 is great because it gets negative signs involved, but it's kinda limited because m needs to equal 1.

- Special Case 2': $m = 1$ still.

  "if$(B_1, \ldots, B_n)$ then toggle C" this is classical reversible - it is its own undo!

  We already saw this when $f(b_1, b_2) = b_1$ AND b_2, and a bunch of other stuff.

  Toggle C is just when $B_1, \ldots, B_n$ is empty (no parameters).

  on the side, adds new qubits, but they'll be cleaned up to 0 at the end.

- General Case: $m \geq 1$

  "toggle $f(B_1, \ldots, B_n)$ onto $C_1, \ldots, C_m$" (XOR)

  classical instruction, and is reversible

  generally the $C_1, \ldots, C_m$ are all set to zero, so basically we keep the n input qubits and map the outputs directly onto m output qubits.

22.1 Exercises

(c) Prove that SeeZee(A,B) is equivalent to the following code:

```
ZeeSee(A,B):
    Hadamard A
    If B then toggle A
    Hadamard A
```

Hint: if you've solved part (b), then you can solve this part using just one or two English sentences.

(d) Using just part (c) plus some basic known facts about quantum instructions, explain why the following code is equivalent to `if B then toggle A`:

```
Hadamard A
Hadamard B
If A then toggle B
Hadamard B
Hadamard A
```

Remark: this is not just a party trick. In some physical implementations of qubits and quantum instructions, it might be physically easy to do `if A then toggle B` but not so easy to physically do `if B then toggle A`. Multiple-qubit operations are always way harder to implement than single-qubit operations. This problem shows that if you can do the former instruction (and also Hadamard), you can use it to build the latter instruction.

22.2 Solutions

(c) SeeZee(A,B) checks if A and B are both 1. ZeeSee has the identical structure, but it swaps bits A and B in the implementation - no matter, since WLOG, bits A and B will be recovered to their initial state unless A and B are both 1, in which case their amplitude on $|11\rangle$ after ZeeSee will be -1. Thus SeeZee and ZeeSee return the same values for any 2-qubit $|AB\rangle$.

(d) Note the middle 3 lines are just SeeZee. Line 1 puts amplitude on both basic states for A, which is undone in line 5 if B is zero (because the middle 3 lines can just be ignored if A or B is zero).
However, when B is 1, then the middle three lines make the amplitude go from 1 on $|11\rangle$ to -1 on $|11\rangle$. Then line 5 doesn't recover the initial state, but will instead make A go from $1 \rightarrow 1$ with positive amplitude (instead of negative like Hadamard usually does) and go from $1 \rightarrow 0$ with negative amplitude (instead of positive like Hadamard usually does).
Then the negative amplitude from $1 \rightarrow 0$ cancels with the other branch where Hadamard doesn't make A into 1, so you end up with 1 amplitude on $|11\rangle$ when the start is $|01\rangle$.
Of course, if you had started with $|11\rangle$, then the same branch effects hold, execpt your initial Hadamard puts negative amplitude on $|11\rangle$ (since it's negative to stay at 1 from 1), making the final result $|01\rangle$.

23 AND/OR/NOT code (for CCode-QCode) (Lecture 7)

Video #023/100 Table of Contents

- Main Case: $m = 1$.
- "if $f(B_1, B_2, \ldots, B_n)$ then toggle C"
- Main Claim: there is a (simple) mechanical procedure that transforms efficient CCode for computing f to efficient quantum code doing the above instruction.

  In "real life" you wouldn't use this procedure. You'd handcraft the quantum code.

  the idea of "efficient" here just means if you have an efficient CCode for f, you can make efficient QCode for f

  if CCode runs in T steps, the AND/OR/NOT code has $O(T^2)$ lines.

Step 1: Convert CCode to Turing Machine code (Church-Turing thesis)

Step 2: Turing Machine code to AND/OR/NOT code

steps 1 and 2 are taught in 15-251 and 15-455 respectively.

23.1 Exercises

(a) Earlier, I asked you to write quantum code to implement "if (A or B) then toggle C" and also "if (not A) then toggle B". I should have asked you to do "if (A xor B) then toggle C". So please now write quantum code for this. (Hint: your solution should be two lines long).

(b) Onto the main problem. Consider the following Boolean function, IsSorted:$\{0,1\}^4 \to \{0,1\}$. It outputs 1 if the four input bits are either in ascending order or descending order; it outputs 0 if they are neither (i.e., "unsorted"), For example, IsSorted(0,0,1,1) = 1 and IsSorted(1,1,1,0) = 1, but IsSorted(0,1,1,0) = 0. (Remark: this is not just some completely random function I made up; it actually plays a minor role in quantum complexity theory, and is sometimes called Ambaini's function).
Write "AND/OR/NOT" code in the style of Lecture for compluting IsSorted(a_1, a_2, a_3, a_4). But don't just blurt it out. First, write a paragraph of English explaining the high-level idea of how your code is going to work. Then write the code, but also put appropriate comments in your code. Finally, for your convenience, you can make it "AND/OR/NOT/XOR" code; that is, you may also use instructions of the form "$w_i = v_j$ XOR v_k".

23.2 Solutions

(a) using previous instructions,

```
if (A or B) toggle C
if (A and B) toggle C
```

(b) though I haven't taken 15-455, I'll take a stab at it. It seems like you can make as many intermediate w variables, AND, NOT, and OR on variables of the same "type".
Let's make some observations first.

- WTS ascending or descending, AKA $(b1 \leq b2 \wedge b2 \leq b3 \wedge b3 \leq b4) \vee b1 \geq b2 \wedge b2 \geq b3 \wedge b3 \geq b4$
- Let's break this down into smaller parts.

- $x \leq y$ is the same as $x == y \vee x < y$
- which is the same as $\neg(x \oplus y) \vee \big((x \oplus y) \wedge (\neg x \wedge y)\big)$
- which by the rule $a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$, becomes
- $\neg(x \oplus y) \vee (x \oplus y)$ and $\neg(x \oplus y) \vee (\neg x \wedge y)$
- which by law of excluded middle becomes $\neg(x \oplus y) \vee (\neg x \wedge y)$

- However, $x \geq y$ is almost the same, so we can reuse the equality part. Except now, we make the other condition $(x \wedge \neg y)$.

`IsSorted(b1, b2, b3, b4)`

w1: NOT b1

w2: NOT b2

w3: NOT b3

w4: NOT b4

- now we have b1 through b4 and all their negations stored.
- Block 1: check if $b1 \leq b2$

w5: b1 XOR b2 $b1 \oplus b2$

w6: NOT w5 $\neg(b1 \oplus b2)$

w7: w1 AND b2 $\neg b1 \wedge b2$

w8: b1 AND w2 $b1 \wedge \neg b2$

w9: w6 OR w7 $b1 \leq b2$

w10: w6 OR w8 $b1 \geq b2$

- now repeat for $b2 \leq b3$:

w11: b2 XOR b3 $b2 \oplus b3$

w12: NOT w11 $\neg(b2 \oplus b3)$

w13: w2 AND b3 $\neg b2 \wedge b3$

w14: b2 AND w3 $b2 \wedge \neg b3$

w15: w12 OR w13 $b2 \leq b3$

w16: w12 OR w14 $b2 \geq b3$

- now repeat for $b3 \leq b4$:

w17: b3 XOR b4 $b3 \oplus b4$

w18: NOT w17 $\neg(b3 \oplus b4)$

w19: w3 AND b4 $\neg b3 \wedge b4$

w20: b3 AND w4 $b3 \wedge \neg b4$

w21: w18 OR w19 $b3 \leq b4$

w22: w18 OR w20 $b3 \geq b4$

- now put it all together:

w23: w9 AND w15 $b1 \leq b2 \leq b3$

w24: w23 AND w21 $b1 \leq b2 \leq b3 \leq b4$

w25: w10 AND w16 $b1 \geq b2 \geq b3$

w26: w25 AND w22 $b1 \geq b2 \geq b3 \geq b4$

24 Reversible Computing (for CCode-QCode)

Video #024/100 Table of Contents

ex: "if IsPalindrome($B_1, \ldots, B_4$) then toggle C" We made AND/OR/NOT code already:
`IsPalindrome(b1, b2, b3, b4)`

w1: b1 AND b4

w2: NOT b1

w3: NOT b4

w4: w2 AND w3

w5: w1 OR w4

w6: b2 AND b3

w7: NOT b2

w8: NOT b3

w9: w7 and w8

w10: w6 OR w9

w11: w5 AND w10

C:= w11

Step 3: take AND/OR/NOT code and compile it to "trashy" quantum code.

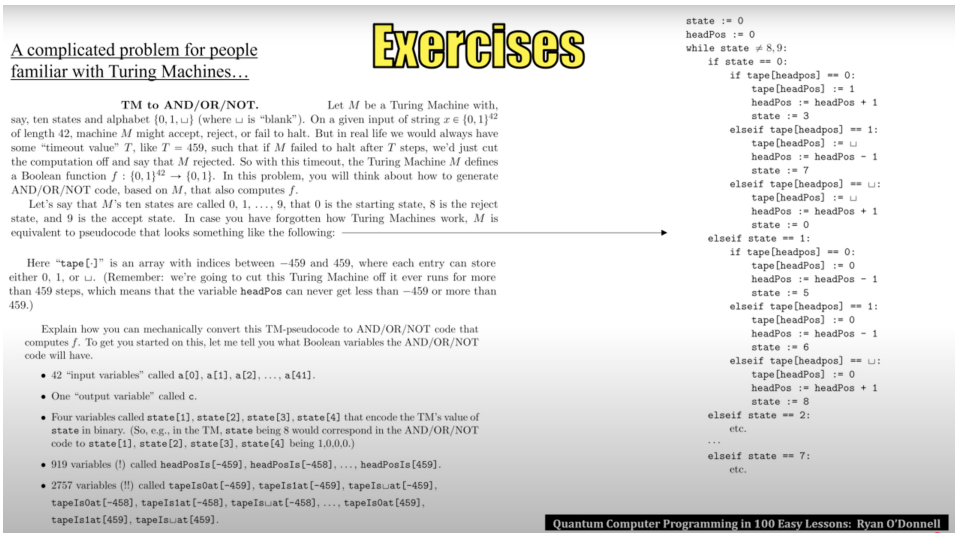
0. `new qubit w1 w2 ... w11` as shown in the AND/OR/NOT code
1. `if (B1 AND B4) then toggle w1`
2. `if (NOT B1) then toggle w2`
3. `if (NOT B1) then toggle w3`
4. `if (w3 AND w3) then toggle w4`
5. `if (w1 OR w4) then toggle w5`
6. `if (b2 AND b3) then toggle w6`
7. `if (NOT b2) then toggle w7`
8. `if (NOT b3) then toggle w8`
9. `if (w7 AND w8) then toggle w9`
10. `if (w6 OR w9) then toggle w10`
11. `if (w5 AND w10) then toggle w11`
12. `if (w11) then toggle C`

The w qubits are workspace qubits (formally, "ancilla" qubits).

We have 16 output qubits from only 5 inputs (B1,B2,B3,B4,C). It's bad to have leftover qubits, it's terrible to leave the "trash" qubits around. Clean up after yourself! Leave the workspace qubits at basic state 0.

To clean up, you "uncompute" everything. Undo all the instructions you've just did! And conveniently, all of the instructions are their own inverses, so just run line 11 again, then line 10 again, then line 9 again, ... back to line 1!

24.1 (COMPLETE AFTER 251) Exercises



A complicated problem for people familiar with Turing Machines. I won't even bother typing it out - I don't know how any of this works!

24.2 Solutions

Notice the TM code is like trashy quantum code.

- if state[1,2,3,4] = 0000 then toggle tapeIs1AtPos[1]
- untoggle headPos[i] and toggle headPos[i + 1]

So mechanically,

- When we see state := i for some i, we set state[1,2,3,4] = i_b where i_b is the binary representation of i
- When we see headPos := i for some i, we set headPos[i] = 1 (AKA toggle it) and set all other headPos variables to 0
- when we see tape[headPos] := x for some $x \in \{0,1,\sqcup\}$, we look for i where headPos[i] = 1 and set tapeIsxAt[i] to 1 and tapeIsyAt[i] to 0 for $y \in \{0,1,\sqcup\}$ such that $y \neq x$.
- for all the if conditioning, we just do the same as if then Toggle from Step 3 above.

Initializing the tape: For all $i \in \{-459, ..., 459\}$, where every boolean variable starts at 0,

- set tapeIs⊔at[i] to 1 (aka do NOT on this variable)
- tapeIs0At[i] and tapeIs1At[i] should all be set to 0.
- headPosIs[i] is set to 0, except for headPosIs[0] which is set to 1 (aka do NOT on this variable)
- then for t from 0 to 41, set tapeIsxAt[t] = 1 if a[t] = x, where $x \in \{0,1,\sqcup\}$

We've finished our initialization, adding the input to the tape and having the correct initial state.

25 Finishing CCode to QCode

Video #025/100 Table of Contents

Our trashy quantum code does
"if IsPalindrome(B1, ..., B4) then toggle C
new qubit w1,..., w11"

Fact: doing `new qubit w1,..., w11` then `extract w1,..., w11` immediately is equivalent to doing nothing. So we should probably append an `extract w1,..., w11` to our trashy quantum code. Then we'll essentially cancel out the `new qubit w1,..., w11`.

Our main claim is complete. If you have classical code, we can convert it into quantum code.

The most general case of CCode to QCode is when you have a Boolean function $f : \{0,1\}^n \to \{0,1\}^m$. We want to winf a function "toggle f(B1,..., Bn) onto C1,..., Cm"

For this kind of function, we can still do the CCode to QCode conversion. We'd just do it one by one for each of the output bits, breaking it down into m cases where there's just one output. Or we can just think of convert our CCode to AND/OR/NOT code and then convert accordingly.

25.1 Exercises

Quantum code for classical reversible instructions. Suppose that $f : \{0,1\}^n \to \{0,1\}^n$ is a Boolean function that happens to be reversible. In other words, it is a bijection, or a permutation on the 2^n possible bitstrings. Let $f^{-1} : \{0,1\}^n \to \{0,1\}^n$ denote the inverse function for f (whos "classical paths diagram" is gotten from f's by reversing the arrows).

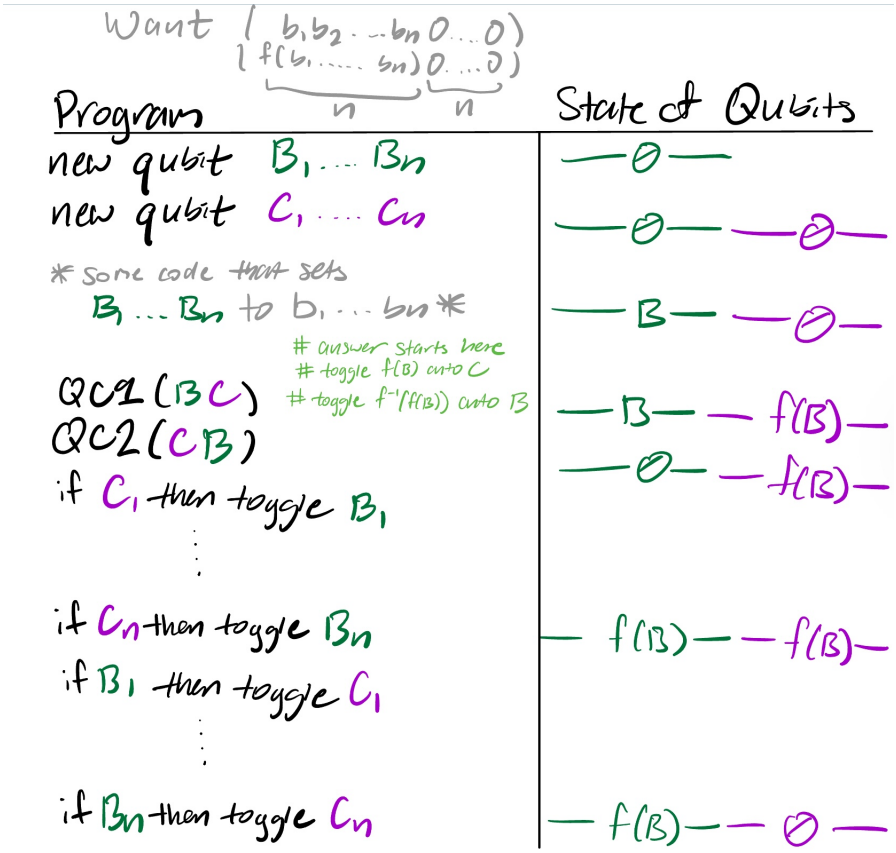
Suppose you happen to have some quantum code $QC1(B_1, \ldots, B_n, C_1, \ldots, C_n)$ that implements the functionality `toggle` $f(B_1, \ldots, B_n)$ `onto` $C_1, \ldots, C_n$. (This is the "standard" way in which the function f may be implemented quantumly). Suppose you also happen to have some quantum code $QC2(B_1, \ldots, B_n, C_1, \ldots, C_n)$ that implements the functionality `toggle` $f^{-1}(B_1, \ldots, B_n)$ `onto` $C_1, \ldots, C_n$.

Now since f happens to be reversible, you know that in principle we ought to be able to get quantum code that simply literally does f to qubits $B_1, \ldots, B_n$ (no "toggling" or anything). In an ideal world, I would now set you the following task: "Using calls to $QC1$ and $QC2$, write some quantum code that simply does f on qubits $B_1, \ldots, B_n$." However, it's not *quite* possible to solve this task.

Instead, solve the following task that's practically identical: Using calls to $QC1$ and $QC2$ (and other basic quantum instructions, if needed), write some quantum code that implements the following spec: "Do f on qubits $B_1, \ldots, B_n$ and also create new qubits called $C_1, \ldots, C_n$." Explain briefly how/why your code works.

Your code may actually internally manipulate the qubits $C_1, \ldots, C_n$ while it's running, but it is critically important that from the perspective of an outsider who simply calls your code, it behaves as if it merely did f on $B_1, \ldots, B_n$ and also did `new qubit` $C_1, \ldots, C_n$.

25.2 Solutions



26 Programming "If f then Minus"

Video #026/100 Table of Contents

We need some positive and some negative amplitudes for quantum computing's whole cancellation stuff to work.

Case $m = 1$ you want `if f(b1...bn) then minus`

- puts amplitude $+1$ on state $|b_1 \dots b_n\rangle$ if $f(b_1 \dots b_n) = 0$
- puts amplitude -1 on state $|b_1 \dots b_n\rangle$ if $f(b_1 \dots b_n) = 1$

Step 1: Get quantum code for `if f(b1...bn) then toggle C`

1. `new qubit C`
2. `toggle C`
3. `Hadamard C` (or `Add&Diff C`)
4. `if f(b1...bn) then toggle C`
5. `Hadamard C` (or `Add&Diff C`)
6. `toggle C`
7. `extract C`

Let's analyze by example (though it says Hadamard, think Add&Diff just to avoid ugly $\sqrt{\frac{1}{2}}$). Say $B_1, \dots, B_n$ in state $|0100\rangle$

1. `new qubit C`

   now state $|B_1 \dots B_n C\rangle = |01000\rangle$ (amplitude $+1$)

2. `toggle C`

   now state $|B_1 \dots B_n C\rangle = |01001\rangle$ (amplitude $+1$)

3. `Hadamard C` (or `Add&Diff C`)

   now state $|B_1 \dots B_n C\rangle = |01000\rangle$ (amplitude $+1$) and $|01001\rangle$ (amplitude -1)

4. `if f(b1...bn) then toggle C`

   Case 1: Assume $f(0100) = 0$. Then we don't toggle at all.

   Case 2: Assume $f(0100) = 0$. Then we toggle. now state $|B_1 \dots B_n C\rangle = |01000\rangle$ (amplitude -1) and $|01001\rangle$ (amplitude $+1$)

5. `Hadamard C` (or `Add&Diff C`)

   Case 1: undoes the Hadamard from Line 3.

   We return to the state $|B_1 \dots B_n C\rangle = |01001\rangle$ (amplitude $+1$)

   Case 2: new amplitude on $|01000\rangle$ is $-1 + 1 = 0$

   new amplitude on $|01001\rangle$ is $-1 + -1 = -2$

   so the new state is -2 amplitude on $|01001\rangle$. We'd normalize here too.

6. Now we get rid of C. `toggle C` (undoes line 2).
7. `extract C` (cancels the hypothetical "`new qubit C`")

Black box this in your mind. Just remember that if we have CCode that does a Boolean, we can do it in QCode.

26.1 Exercises

This refers to the "IsSorted" function from the exercises of Video #023/100.

Write quantum code for if "IsSorted(A_1, A_2, A_3, A_4) then minus". You don't have to justify your solution, as long as it's correct. For your convenience, you may treat "if (A or B) then toggle C" and "if (not A) then toggle B" and "if (A xor B) then toggle C" as "basic pseudocode isntructions"; i.e., you do not have to break them down further.

26.2 Solutions

1. new qubit $w_1, \ldots, w_{11}$
2. if (NOT) A_1 then toggle w_1
3. if (NOT) A_2 then toggle w_2
4. if (NOT) A_3 then toggle w_3
5. if (NOT) A_4 then toggle w_4
6. if A_1 XOR A_2 then toggle w_5

   $\vdots$

I really don't want to write all this out. It's just copying the AND/OR/NOT code from the exercise for Video 23, and it's tedious.

27 Loading + - 1 truth tables into the state (Lecture 8)

Video #027/100 Table of Contents

Recall the first part of the "paradigm":

1. Prepare the uniform superposition on n qubits (amplitude $+1$ on all 2^n basic states)
2. Apply "if f then minus/toggle" getting all 2^n input/output pairs encoded into states amplitudes

Claim: this code performs step 1. O'Donnell proves the $n = 3$ case as an example; I can't be asked to draw the cube so I'll just typeset the $n = 2$ case.

0. new qubit B1...Bn $|00\rangle$
1. Add&Diff B1 $|00\rangle$ amplitude 1, $|10\rangle$ amplitude 1.

   ⋮

n. Add &Diff Bn $|00\rangle$ amplitude 1, $|01\rangle$ amplitude 1, $|10\rangle$ amplitude 1, $|11\rangle$ amplitude 1

Now we look at step 2. Say we have a function $f : \{0,1\}^n \to \{0,1\}$ and we have converted it from CCode to QCode. After line 2 (run the if f then toggle), we can reach a new state where $|b_1, \ldots, b_n\rangle$ has amplitude $+1$ if $f(b_1, \ldots, b_n) = 0$ and amplitude -1 if $f(b_1, \ldots, b_n) = 1$. We've encoded the truth table of f in ± 1 notation into the state of the qubits. True is -1, false is $+1$.

Notice that right now, all of the basic states have equal probability. Thus, we need to use Hadamard to have some things cancel and some things not cancel!

27.1 Exercises

Your annoying but trustworthy friend writes a quantum subroutine called Enigma, which operates on two qubits. Your friend promises you that Enigma(A,B) is just a single line long, and that line is one of the following four possibilities:

1. if A then toggle B
2. if not A then toggle B
3. toggle B
4. no-op

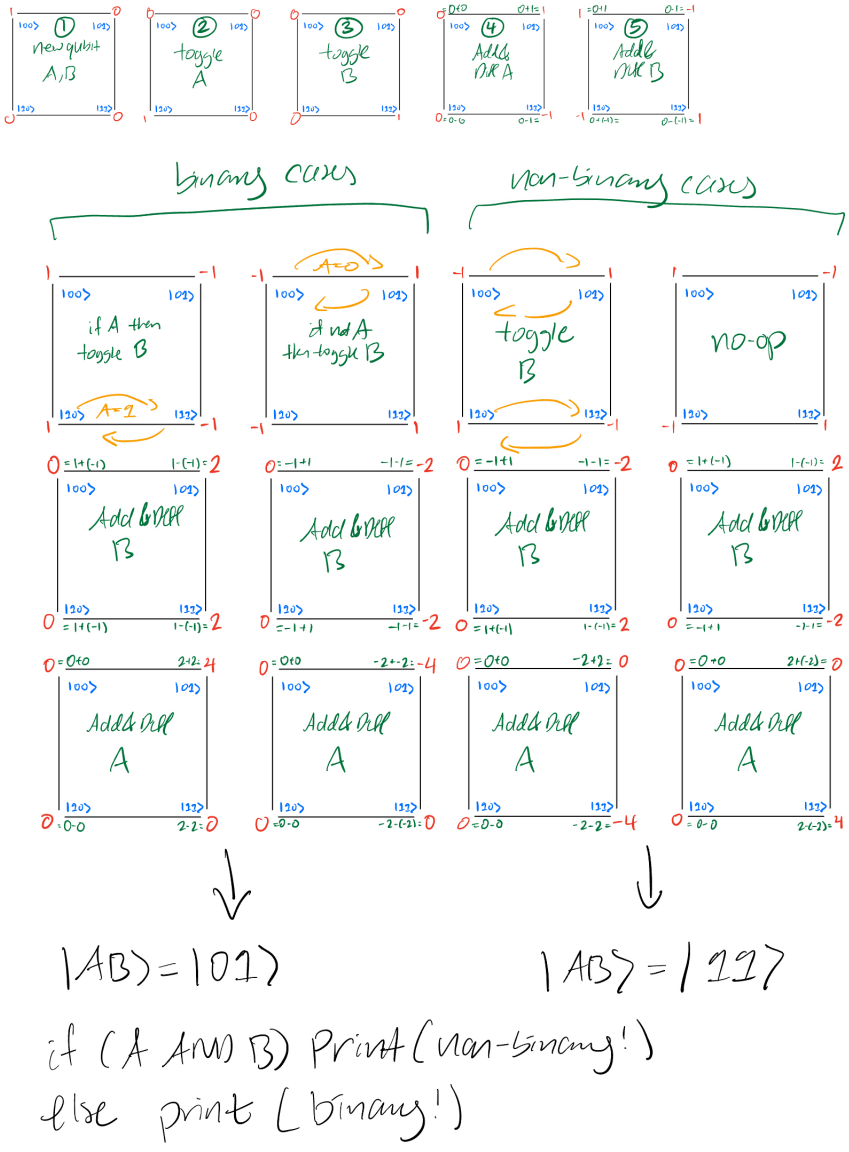
Let's categorize the first two possibilities above as "binary", and the last two possibilities above as "non-binary".

Write some code (quantum code + some "classical post-processing") that makes a *single call* to the Enigma subroutine and outputs either "Binary!" or "Non-Binary!". Your code should have the probability that it has a 100% probability of being correct.

27.2 Solutions

1. new qubit A B
2. toggle A
3. toggle B
4. Add&Diff A
5. Add&Diff B
6. Enigma (A, B)
7. Add&Diff B
8. Add&Diff A
9. extract A B

print (A AND B ? non-binary! : binary!)



28 Avg & Dev All (Hadamard Transform)

Video #028/100 Table of Contents

Step 3: "Do Fourier Transform" — just do "Hadamard All" (or Add&Diff All to use unnormalized states), or we are going to introduce a new thing — Avg&Dev All

Somehow, take a signal and transform it into another signal. The Hadamard transform (Hadamard All) is a type of Fourier transform. Transforms the amplitudes to somehow represent the frequencies of the original data.

Avg&Dev A is similar to Add&Diff A - it's just Hadamard A, then multiply all the amplitudes by $\sqrt{\frac{1}{2}}$. Basically, Avg&Dev leaves amplitudes of $\frac{1}{2}$ and $-\frac{1}{2}$, whereas Add&Diff leaves amplitudes of 1 and -1.

Recall how these operations got their names: given starting amplitudes x on $|0\rangle$ and y on $|1\rangle$,

- Hadamard leaves $\sqrt{\frac{1}{2}}(x+y)$ on $|0\rangle$ and $\sqrt{\frac{1}{2}}(x-y)$ on $|1\rangle$
- Add&Diff leaves $x+y$ (Add) on $|0\rangle$ and $x-y$ (Diff) on $|1\rangle$
- Avg&Dev leaves $\frac{x+y}{2}$ (Arithmetic Avg) on $|0\rangle$ and $\frac{x-y}{2}$ (Deviation from Avg) on $|1\rangle$

Why do we introduce this new instruction? It's the undo of Add&Diff! (Avg&Dev have opposite normalization constants!) It makes analysis very easy: given an even number of Hadamard instructions on a qubit, we can make half of them into Add&Diff and half into Avg&Dev.

When we prepare the uniform superposition state, we use Add&Diff. Then at the end, we can just use Avg&Dev which will return us to a normalized state!

Q: What happens to amplitudes if you do Avg&Dev to all of them?
A: Partial answer: on all zeros, is the average of all the ± 1 truth table entries (assuming you've already performed the preceding steps). It's generally true that Avg&Dev to all qubits (which are currently in a superposition state) will just average them.

Again, O'Donnell proves by example up to $n=3$. True for $n=1$ by defn. of Avg&Dev. So let's look at the $n=2$ case.

- start in state (a,b,c,d) for $|00\rangle, |01\rangle, |10\rangle, |11\rangle$
- Let's Avg&Dev(B1). Then we get $(\frac{a+b}{2}, \frac{a-b}{2}, \frac{c+d}{2}, \frac{c-d}{2})$
- Let's Avg&Dev(B2). Then we get $\left(\frac{\frac{a+b}{2}+\frac{c+d}{2}}{2}, \frac{\frac{a-b}{2}-\frac{c+d}{2}}{2}, \frac{\frac{a+b}{2}-\frac{c-d}{2}}{2}, \frac{\frac{a-b}{2}-\frac{c-d}{2}}{2}\right) = \left(\frac{a+b+c+d}{4}, \frac{a-b-c-d}{4}, \frac{a+b-c+d}{4}, \frac{a-b-c+d}{4}\right)$

28.1 Exercises

Ross runs a machine learning algorithm that produces a "classifier", which is some AND/OR/NOT code C with n input bits ("features") called $x_1, x_2, \ldots, x_n$ and 1 output bit ("classification"). Unfortunately, it's very hard to interpret what C is doing, so Ross wants to try to build a single "decision tree" that mimics C. A common way to get started (in the "ID3 algorithm") is to try to find an inpu bit $j \in \{1,2,\ldots,n\}$ that has a large "Expectation Gain", where

$$\text{ExpectationGain}(j) = \text{average}\left\{C(x) : x \in \{0,1\}^n, x_j = 1\right\} - \text{average}\left\{C(x) : x \in \{0,1\}^n, x_j = 0\right\}$$

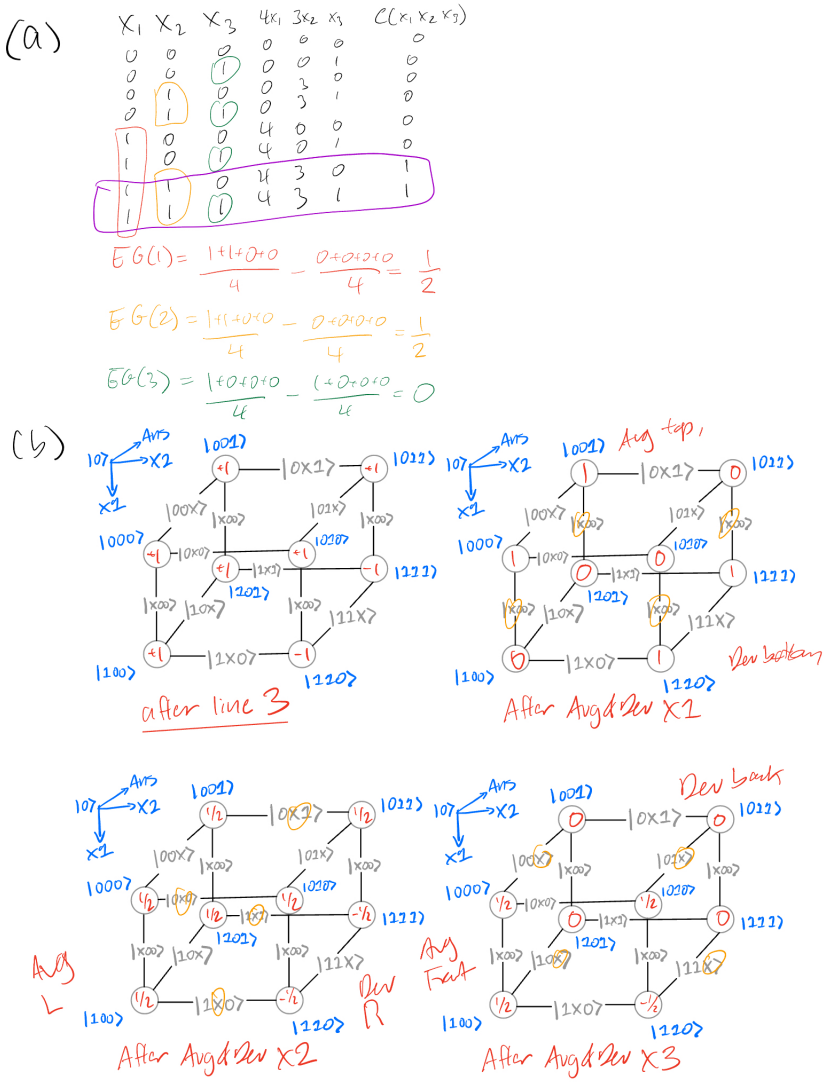
(a) As an example, suppose $n = 3$ and $C(x_1, x_2, x_3)$ happens to have the following behavior: it outputs 1 if and only if $4x_1 + 3x_2 + x_3 > 6$ (e.g., $C(1, 0, 1) = 0$ because $5 \not> 6$.) Write the truth table for C, and compute the three values ExpectationGain(1), ExpectationGain(2), and ExpectationGain(3).

(b) Consider the following quantum code:

1. new qubit $X_1, X_2, \ldots, X_n$
2. Add&Diff on $X_1, X_2, \ldots, X_n$
3. if $C(X_1, X_2, \ldots, X_n)$ then minus
4. Avg&Dev on $X_1, X_2, \ldots, X_n$

For the particular $n = 3$ example from part (a), illustrate how the state's amplitudes change throughout the course of the above code by drawing those "cubical diagrams" like we do in lecture. You only have to draw four diagrams: one for the state after line 3 of the code, one after Avg&Dev on X_1, one after Avg&Dev on X_2, and one after Avg&Dev on X_3 (i.e., at the end of the code). As long as your diagrams are correct, you don't need to add any further justitification.

28.2 Solutions



29 Quantum-Bias Busting (Deutsch — Jozsa)

Video #029/100 Table of Contents

Question: Suppose we load the ± 1 notation truth table of $f : \{0,1\}^n \to \{0,1\}$ into our state and then do Hadamard transform. What does it "mean" if the resulting amplitude on $|000\ldots0\rangle$ is 0?
Answer: it means that the truth table of f is unbiased (exactly equal number of 0s and 1s).

Conversely, if f's truth table is biased, then the final amplitude on $|000\ldots0\rangle$ is 1. As a consequence, suppose at the end, you "extract all" (given you've converted f's CCode to QCode using this paradigm).

- If f's truth table is unbiased, then the probability of getting $|000\ldots0\rangle$ is 0.
- If f's truth table is biased, then the probability of getting $|000\ldots0\rangle$ is nonzero.

So if we see $|000\ldots0\rangle$ after extract all, we know f's truth table is biased. It's probabalistic though, not deterministic.

The **Quantum Bias-Busting Algorithm** (formally, the **Deutsch - Jozsa Algorithm**): Given as input some AND/OR/NOT code computing $f : \{0,1\}^n \to \{0,1\}$,

1. Convert to quantum code for "if f then minus"
2. new qubit B1 ... Bn
3. Hadamard All (Add&Diff)
4. if f then minus
5. Hadamard All (Avg&Dev)
6. extract all
7. if the display is "000...0", output biased! else no comment.

Properties:

1. This algorithm has no false positives. It's literally impossible to output biased! when f's truth table is unbiased.

   But the algorithm that always outputs no comment also never has false positives.
2. This algorithm has at least some positive power: if f's truth table is biased, the probability of outputting biased! is not zero.
   Comes from statistical definition of power: Given the null hypothesis is false, the algorithm with reject the null hypothesis (AKA return biased!) with nonzero probability (just watched a YouTube video from Very Normal on power!)
3. This quantum algorithm is efficient! The total number of operations is in $O(n)$ given n qubits. "if f then minus" is within $O(\#$ of AND/OR/NOT code lines)
4. This algorithm is practically useless (though O'Donnell just ends the video here, I suspect it's because it's probabilistic, which means its power can be very low).

29.1 Exercises

This follows up on the exercise from Video #028/100.

(c) Prove that in the general-n case, at the end of this code, the amplitude on $|000\ldots01\rangle$ is exactly ExpectationGain(n). (Hint, you'll probably find it helpful to define $S(x) = 1 - 2C(x)$, and reexpress ExpectationGain(n) in terms of $S(x)$ rather than $C(x)$).

(d) Explain why, in general, the final amplitude on $|0\ldots010\ldots0\rangle$ (where the 1 is in the jth position) is equal to ExpectationGain(j). Your explanation should start by saying, "As we know (previous exercise), since Avg&Dev on X_1 ... Avg&Dev on X_n operate on separate qubits, it does not matter what order they are applied in. In particular, it would be the same if we did "Avg&Dev on X_j" last. So..."

29.2 Solutions

(c) Let $S(x) = 1 - 2C(x)$ (this puts everything into ±1 notation truth table).

$$\begin{aligned}
\text{ExpectationGain}(n) &= \text{average}\left\{C(x) : x \in \{0,1\}^n, x_n = 1\right\} - \text{average}\left\{C(x) : x \in \{0,1\}^n, x_n = 0\right\} \\
&= \frac{1}{2^{n-1}}\left(\sum_{x_n=1} C(x) - \sum_{x_n=0} C(x)\right) = \frac{1}{2^{n-1}}\left(\sum_{x_n=1} \frac{1-S(x)}{2} - \sum_{x_n=0} \frac{1-S(x)}{2}\right) \\
&= \frac{1}{2^n}\left(\sum_{x_n=1} 1 - S(x) - \sum_{x_n=0} 1 - S(x)\right) = \frac{1}{2^n}\left(\sum_{x_n=1} -S(x) - \sum_{x_n=0} -S(x)\right) \\
&\qquad\qquad \text{(same \# cases when } x_j = 1 \text{ versus } x_j = 0) \\
&= \frac{1}{2^n}\left(\sum_{x_n=0} S(x) - \sum_{x_n=1} S(x)\right) \qquad \text{(get rid of double negative sign)} \\
&= \text{weighted average of } \pm1 \text{ notation truth table values}
\end{aligned}$$

At the end of the code, we know $|000\ldots00\rangle$ is the average of all the $S(x)$ values by definition. Then

$$\text{amplitude for } |000\ldots00\rangle = \frac{1}{2^n}\left(\sum_{x_n=0} S(x) + \sum_{x_n=1} S(x)\right)$$

But when we do Avg&Dev(X_n), the Avg is onto $|000\ldots00\rangle$, and the Dev is onto $|000\ldots00\rangle$. Thus instead of adding, we do Deviation (which is subtracting) for the amplitude on $|000\ldots01\rangle$, which is why

$$\text{amplitude for } |000\ldots01\rangle = \frac{1}{2^n}\left(\sum_{x_n=0} S(x) - \sum_{x_n=1} S(x)\right) = \text{ExpectationGain}(n)$$

(d) As we know (from Video #13/100's exercises), since Avg&Dev on X_1 ... Avg&Dev on X_n operate on separate qubits, it does not matter what order they are applied in. In particular, it would be the same if we did "Avg&Dev on X_j" last. So since

$$\text{amplitude for } |000\ldots00\rangle = \frac{1}{2^n}\left(\sum_{x_j=0} S(x) + \sum_{x_j=1} S(x)\right)$$

Then the Dev for the step when we do Avg&Dev(X_j) gives us the amplitude for $|0\ldots010\ldots0\rangle$ (where the 1 is in the jth position) is

$$\frac{1}{2^n}\left(\sum_{x_j=0} S(x) - \sum_{x_j=1} S(x)\right) = \text{ExpectationGain}(j)$$

30 Is Quantum Bias-Busting Cool?

Video #030/100 Table of Contents

Why is the Deutsch-Jozsa Algorithm useless? (like I said in the notes for #Video 029/100), perhaps the truth table is biased, but the truth table is massive and *barely* biased. Then the average of the ± 1 notation truth table values is extremely small.

e.g, perhaps the truth table is $2^9 99 + 1$ 0s and $2^9 99 - 1$ 1s. Then the probability of seeing all zeros and the algorithm outputs biased! is $\frac{(2^{999}+1)(1)+(2^{999}-1)(-1)}{2^{1000}} = \frac{2}{2^{1000}} = \frac{1}{2^{999}}$.

Then why are we still interested in this algorithm?

Question: Could their be any classical probabalistic algorithm with the first 3 properties of Deutsch-Jozsa? (no false pos, has nonzero power, is efficient). Seems impossible that a classical algorithm has none of these properties. Computational complexity researchers thought about this question about 1990 (before quantum computing had been known of).

Mitori Ogihara's Theorem: No such classical algorithm exists assuming $P \neq NP$. Then quantum computing can do things classical computing can't do, *but we don't know for sure unless* $P \neq NP$. (From 15-455, this basically means $NP^{ND} \neq coNP^{ND}$).

30.1 (COMPLETE AFTER 15-455) Exercises

Only tackle these if you've had a course in computational complexity theory...

For now, I will not typeset these.

30.2 Solutions

31 Bitmasked XOR Functions (Lecture 9)

Video #031/100 Table of Contents

`Quantum TogglesDetective`

1. `new qubit` $X_1, \ldots, X_6, Ans$
2. `toggle` Ans
3. `Hadamard all`
4. `MysteryToggles`
5. `Hadamard all`
6. `extract all`

`MysteryToggles`

1. if X_1 then toggle Ans

#2. if X_2 then toggle Ans

3. if X_3 then toggle Ans
4. if X_4 then toggle Ans
5. if X_5 then toggle Ans

#6. if X_6 then toggle Ans

Claim: MysteryToggles is of the form "if $f(X_1, \ldots, X_6)$" for a certain $f : \{0,1\}^6 \to \{0,1\}$. In our example, it's the function $f(x_1, \ldots, x_6) = x_1 \oplus x_3 \oplus x_4 \oplus x_5 = XOR_{10110}$ function, which is a bitmasked XOR function.

definition: the (bitmasked) $XOR_{b_1 \ldots b_n}$ function maps n bits to 1 bit as follows:

- $XOR_{b_1 \ldots b_n}(x_1 \ldots x_n)$ = "XOR of all x_j's where $b_j = 1''$ $= (b_1 \wedge x_1) \oplus \cdots \oplus (b_n \wedge x_n)$
- Quirk: the b's and x's are swappable: $XOR_{b_1 \ldots b_n}(x_1 \ldots x_n) = XOR_{x_1 \ldots x_n}(b_1 \ldots b_n)$
- $XOR_{111\ldots11}(x_1 \ldots x_n) = XOR$ of all $x_1 \ldots x_n$
- $XOR_{000\ldots00}(x_1 \ldots x_n) = 0$
- $XOR_{100\ldots00}(x_1 \ldots x_n) = x_1$

Now let's write an equivalent TogglesDetective: `Quantum TogglesDetective`

1. `new qubit` $X_1, \ldots, X_6$
2. `Add&Diff all` uniform superposition state

   now we need to do "if XOR_{101110} then minus"
3. `new qubit` Ans
4. `toggle` Ans
5. `Hadamard` Ans
6. if XOR_{101110} then toggle Ans
7. `Hadamard` Ans
8. `toggle` Ans

9. `extract` Ans

lines 3-9 are recipe for "if f then minus" Video #026/100

10. `Avg&Dev all`
11. `extract all`

Now we've cleaned up the answer qubit! This way, we don't have the leftover extra qubit.

31.1 Exercises

- What is the value of $XOR_{1111}(1111)$?
- What is the value of $XOR_{1011}(1100)$?
- What is the value of $XOR_{1100}(1011)$?
- Consider the function $f : \{0,1\}^3 \to \{0,1\}$ defined by $f(x_1, x_2, x_3) = x_3$. This is equivalent to $XOR_{b_1 b_2 b_3}$ for which string $b_1 b_2 b_3$?

31.2 Solutions

- it's just 1 xor 1 xor 1 xor 1 = 0
- it's 1 xor 0 xor 0 = 1
- it's the same as the previous since the bitmasked XOR functions are swappable, also 1
- $b_1 b_2 b_3 = 001$

32 Correlation

Video #032/100 Table of Contents

Seemingly, if you start a state with amplitudes being the ± 1 truth table of some bitmasked XOR function $XOR_{b_1,b_2,\ldots,b_n}$ then do Hadamard transfor, the result is $|b_1 b_2 \ldots b_n\rangle$.

Theorem: Suppose state of n qubits is the ± 1 truth table of some Boolean function $f : \{0,1\}^n \to \{0,1\}$, and you do Hadamard transform (Avg&Dev all). Then the resulting amplitude on $|b_1 b_2 \ldots b_n\rangle$ is the **correlation** of f and $XOR_{b_1,b_2,\ldots,b_n}$.

The **correlation** of two Boolean functions $f, g : \{0,1\}^n \to \{0,1\}$ is

$$\text{corr}(f,g) = \text{“fraction of } x \in \{0,1\}^n \text{ where } f(x) = g(x) - \text{fraction of } x \in \{0,1\}^n \text{ where } f(x) \neq g(x)''$$

Facts: $\text{corr}(f,g)$ is a number between -1 and 1.

- $\text{corr}(f,g) = 1$ iff $f = g$
- $\text{corr}(f,g) = -1$ iff $f = \neg g$
- $\text{corr}(f,g) = 0$ iff f, g agree on half of all 2^n inputs.

Question: What is $\text{corr}(f, XOR_{000\ldots00})$?
Answer: $XOR_{000\ldots00}$ is a function that's always zero. So corr is fraction of inputs x where $f(x) = 0$ minus the fraction where $f(x) = 1$. Measures the "biasedness" of f. $\text{corr}(f) = 0$ iff f is unbiased. But this is the same as the weighted average for x, where we add weight 1 if $f(x) = 0$ and -1 if $f(x) = 1$. This is then the same as the average for x when $f(x) \in \pm 1$ notation. $\text{corr}(f, XOR_{000\ldots00})$ is the average of f's ± 1 truth table values (Video #028/100).

Indirect proof: we know we'll get back a normalized state after TogglesDetective, and we know we'll get amplitude 1 on the $b_1 \ldots b_n$ - the correlation of f with itself is 1. But since we have a normalized state, every other state has amplitude 0.

32.1 Exercises

Correlation of bitmasked XORs. If one starts from an n-qubit state where all the amplitudes are the $\pm$ truth table of the $XOR_{b_1 b_2 \ldots b_n}$ function, and then one does "Avg&Dev all", the resulting state's amplitude on $|b_1 b_2 \ldots b_n\rangle$ is $\text{corr}(XOR_{b_1 b_2 \ldots b_n}, XOR_{b_1 b_2 \ldots b_n}) = 1$. Given that the resulting state is also normalized, it follows that the amplitudes on other $2^n - 1$ basic states must be 0 (because the sum of the squares of the amplitudes must be 1, and there is already a contribution of 1^2 from $|b_1 b_2 \ldots b_n\rangle$). It follows that for any string $c_1 c_2 \ldots c_n \in \{0,1\}^n$ not equal to $b_1 b_2 \ldots b_n$, we must have

$$\text{corr}(XOR_{b_1 b_2 \ldots b_n}, XOR_{c_1 c_2 \ldots c_n}) = 0$$

However, that is an extremely roundabout argument. Give a *direct* proof of the above equation.

32.2 Solutions

Proof. Fix $B = b_1 b_2 \ldots b_n, C = c_1 c_2 \ldots c_n \in \{0,1\}^n$,
where $B = b_1 b_2 \ldots b_n \neq c_1 c_2 \ldots c_n = C$. WTS $\text{corr}(XOR_{b_1 b_2 \ldots b_n}, XOR_{c_1 c_2 \ldots c_n}) = 0$.

Observations:

- XOR with 0 is a no-op.
- $B = b_1 b_2 \dots b_n \neq c_1 c_2 \dots c_n = C$
- $b_1 \oplus \dots \oplus b_n = (b_1 \wedge x_1) \oplus (b_1 \wedge \neg x_1) \oplus \dots \oplus (b_n \wedge x_n) \oplus (b_n \wedge \neg x_n) = XOR_B(x) \oplus XOR_B(\neg x)$ for any input $x \in \{0,1\}^n$.
- $(a \wedge b) \oplus (a \wedge c)$ is the same as $(a \wedge b \wedge \neg c) \vee (a \wedge \neg b \wedge c)$
- $b_1 \oplus \dots \oplus b_n = (b_1 \wedge x_1) \oplus (b_1 \wedge x_n) \oplus \dots \oplus (b_n \wedge x_n) \oplus (b_n \wedge x_1) = XOR_B(x) \oplus XOR_B(\neg x)$ for any input $x \in \{0,1\}^n$.

By the defn of corr, it suffices to prove that every input x where $XOR_{b_1 b_2 \dots b_n}$ and $XOR_{c_1 c_2 \dots c_n}$ agree is coupled with an input y where $XOR_{b_1 b_2 \dots b_n}$ and $XOR_{c_1 c_2 \dots c_n}$ disagree.

Fix $x \in \{0,1\}^n$ such that $XOR_{b_1 b_2 \dots b_n}(x) \neq XOR_{c_1 c_2 \dots c_n}(x)$. Assume the number of 1's in B differs from the number of 1's in C by magnitude $k \in \mathbb{N}$.

Case 1. k is odd.

$\implies b_1 \oplus \dots \oplus b_n \neq c_1 \oplus \dots \oplus c_n$

$\implies XOR_B(x) \oplus XOR_B(\neg x) \neq XOR_C(x) \oplus XOR_C(\neg x)$

$\implies XOR_B(\neg x) = XOR_C(\neg x)$ since we fixed $XOR_{b_1 b_2 \dots b_n}(x) \neq XOR_{c_1 c_2 \dots c_n}(x)$. Otherwise, we'd get that B and C have the same number of 1's, contradicting the case assumption.

$\implies XOR_B, XOR_C$ agree at x and disagree at $\neg x$.

Case 2. k is even.

Then B and C differ in at least 2 places. Choose 2 of these positions $1 \leq i, j \leq n$.

Since $XOR_B(x) \neq XOR_C(x)$, then there is some position $1 \leq k \leq n$ where $x_k = 1$ and $b_k \neq c_k$ (if there wasn't, then the two bitmasked XOR functions would agree). Choose the first such position k, and flip it to zero. Then this new bitstring y, where $y = x$ except at position k, has the property $XOR_B(y) = XOR_C(y)$.

Since disagreements at bitstring x are coupled with disagreements at another bitstring, there are equal number of agreeements and disagreements, so the two bitmasked XOR functions agree on half the possible n-length bitstrings and thus have correlation zero. □

33 Hadamard Transform: how it works

Video #033/100 Table of Contents

Notation: $\tilde{g} : \{0,1\}^n \to \{\pm 1\}$ for ± 1 notation for Boolean output

Lemma 1: Given $g : \{0,1\}^n \to \{0,1\}$, then $\text{corr}(f, XOR_{b_1 \ldots b_n}) = \text{avg}_{x \in \{0,1\}^n} \left\{ \tilde{f}(x) \cdot \tilde{XOR}_{b_1 \ldots b_n}(x) \right\}$

Lemma 2: These two snippets are equivalent:

```
Snippet 1
```

1. `Avg&Dev A`
2. `toggle A`

```
Snippet 2
```

1. `if A then minus`
2. `Avg&Dev A`

We will prove the theorem from the previous video:

Theorem: Suppose the state of n qubits is the ± 1 truth table of $f : \{0,1\}^n \to \{0,1\}$, and you do Hadamard transform (Avg&Dev all). Then the resulting amplitude on $|b_1 \ldots b_n\rangle$ is the correlation of f with $XOR_{b_1 \ldots b_n}$.

Proof. WTS if we start in state "amplitude $\tilde{f}(x)$ on $|x\rangle \, \forall x \in \{0,1\}^n$" and do "Avg&Dev all", resulting amplitude on $|b_1 \ldots b_n\rangle$ is $\text{avg}_x\{\tilde{f}(x) \cdot \tilde{XOR}_{b_1 \ldots b_n}\}$.

We'll show: for any $g : \{0,1\}^n \to \mathbb{R}$, starting from amplitude $g(x)$ on $|x\rangle \, \forall x$ and doing Avg&Dev all, resulting amplitude on $|b_1 \ldots b_n\rangle$ is $\text{avg}_{x \in \{0,1\}^n}\{g(x) \cdot \tilde{XOR}_{b_1 \ldots b_n}(x)\}$.

Proof by example ($n = 3$, $b_1 b_2 b_3 = 101$): starting from state g, we do Avg&Dev x_1, Avg&Dev x_2, Avg&Dev x_3. To reason about final amplitude on $|101\rangle$, we'll toggle X_1 and toggle X_3, moving old amplitude on $ket101$ to $|000\rangle$.

Let's rearrange the code (recall ops on separate qubits can be done in any order)

1. Avg&Dev X1
2. toggle X1
3. Avg&Dev X2
4. Avg&Dev X3
5. toggle X3

By Lemma 2,

1. if X1 then minus
2. Avg&Dev X1
3. Avg&Dev X2
4. if X3 then minus
5. Avg&Dev X3

Rearrange again!

1. if X1 then minus
2. if X3 then minus
3. Avg&Dev X1

4. Avg&Dev X2
5. Avg&Dev X3

By Video #028/100, the final amplitude on $|000\rangle$ is the average of all amplitudes after line 2. It remains to show that lines 1-2 change amplitude on $|x_1x_2x_3\rangle$ from $g(x_1x_2x_3)$ to $g(x_1x_2x_3) \cdot X\tilde{O}R_{101}(x_1x_2x_3)$. Which is true because that's what if X1 then minus and if X3 then minus does - it multiplies amplitudes on $|x_1x_2x_3\rangle$ by -1 iff an odd number of $\{x_1, x_3\}$ is 1 iff $XOR_{101}(x_1x_2x_3) = 1$. It multiplies by 1 otherwise. □

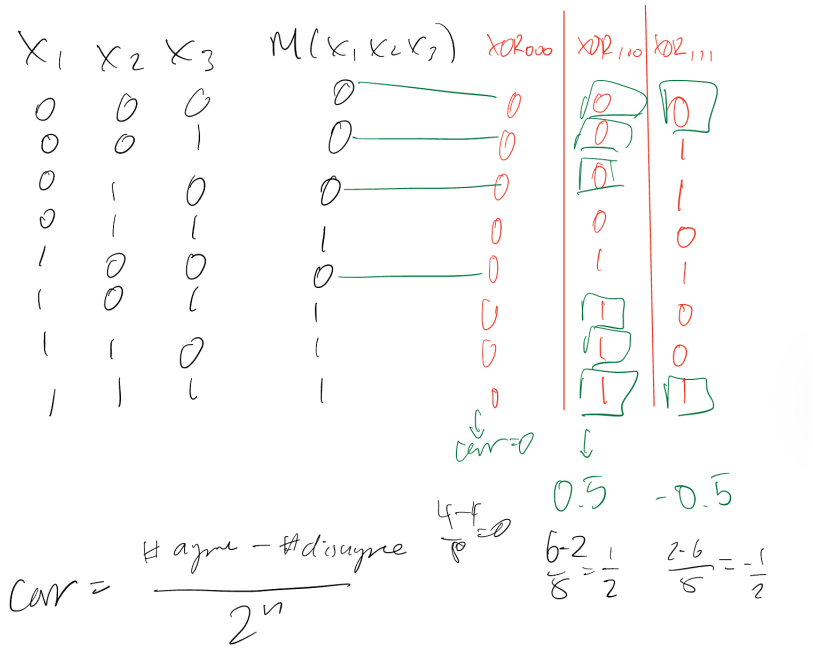
33.1 Exercises

Let $M : \{0,1\}^3 \to \{0,1\}$ denote the "Majority" function of three bits.

- What is $\text{corr}(M, XOR_{000})$?
- What is $\text{corr}(M, XOR_{100})$?
- What is $\text{corr}(M, XOR_{111})$?

33.2 Solutions

- 0
- 0.5
- -0.5



34 Summarizing all quantum algorithms (Lecture 10)

Video #034/100 Table of Contents

Summary: suppose $f : \{0,1\}^n \to \{0,1\}$ is computable efficiently, classically. Then the "Hadamard (Fourier) Transform Paradigm" and extract all lets a quantum computer efficiently see a string $|b_1 \dots b_n\rangle$ with probability $\left(\text{corr}(f, XOR_{b_1 \dots b_n})\right)^2$. What is this good for?

We used this for MysteryToggles, where f is precisely $XOR_{b_1 \dots b_n}$ for a mystery bitstring. Really specific scenario. Reduces from quadratic to linear time.

We used this for Bias Busting: we're interested in $\text{corr}(f, XOR_{00\dots0})$. Impractical though, could be super small probabilities.

MysteryPeriod (aka Simon's Problem): given classical code computing $f : \{0,1\}^n \to \{0,1\}$, somehow you know there's a mystery bitstring $b_1 b_2 \dots b_n \in \{0,1\}^n$ such that $f(x_1 \oplus b_1, x_2 \oplus b_2, \dots x_n \oplus b_n) = f(x_1, x_2, \dots, x_n) \forall x \in \{0,1\}^n$ (and no other two f outputs are equal). Goal: find $b_1 b_2 \dots b_n$.

1993, UToronto, Dan Simon - Jill Brassard (supervisor) told him to look at some quantum computing thing. Tried to prove that quantum computing is useless. Realized that this problem is classically computable in at best exponential time, but there's a super efficient quantum algorithm. This is a contrived problem that doesn't really show up in real life, but it's solvable using the Hadamard Transform Paradigm in quadratic time! Next part of the story: Shor, 1994 reads Simon's paper and figures out how to factor n-bit numbers efficiently, quantumly. Kitaev, 1995: heard of the existence of Shor's algorithm, ended up finding a different such algorithm for factoring efficiently (though they're actually the same, but it uses a different quantum paradigm: Phase Estimation, which is basically Rotation Estimation since we try to ignore complex numbers). Grover, 1996: $\sqrt{2}^n$ time quantum algorithm for SAT, comes from the Rotation Estimation paradigm - geometric viewpoint on quantum algorithms.

34.1 Exercises

You will recall (e.g., from the previous exercise) that if one starts from an n-qubit state where the amplitudes are the ± 1 truth table of the $XOR_{b_1 \dots b_n}$ function, and then one does "Avg&Dev all", the resulting state is $|b_1 \dots b_n\rangle$.

(a) let $|\chi_b\rangle$ denote the 2^n-dimensional vector that looks like $(\sqrt{\frac{1}{2}})^n$ times the ± 1 truth table of $XOR_{b_1 \dots b_n}$. Let H_n denote the $2^n \times 2^n$ unitary matrix corresponding to the n-qubit instruction "Hadamard all". Explain why $H_n |\chi_b\rangle = |b_1 \dots b_n\rangle$.

(b) Explain why $H_n |b_1 \dots b_n\rangle = |\chi_b\rangle$.

(c) Deduce that if A_n denotes the matrix corresponding to the n-qubit operation Avg&Dev all, then the $|b_1 \dots b_n\rangle$ labeled column of A_n looks like the ± 1 truth table of $XOR_{b_1 \dots b_n}$ times $(\frac{1}{2})^n$.

(d) Deduce from this that the $|b_1 \dots b_n\rangle$-labeled row of A_n also looks like the ± 1 truth table of $XOR_{b_1 \dots b_n}$ times $(\frac{1}{2})^n$.

(e) Using just part (d) and the nature of matrix-vector multiplication, give an alternative proof of the following theorem from Lecture 9: if $f : \{0,1\}^n \to \{0,1\}$ and one starts with an n-qubit state whose amplitudes are the ± 1 truth table values of f, and then one performs Avg&Dev all, the resulting amplitude on $|b_1 \dots b_n\rangle$ is equal to $\text{corr}(f, XOR_{b_1 \dots b_n})$.

34.2 Solutions

(a) Use the theorem from the problem statement. $|\chi_b\rangle$ is the n-qubit state where the amplitudes are the ± 1 truth table of the $XOR_{b_1 \dots b_n}$ function, with multiplying factor $(\sqrt{\frac{1}{2}})^n$. Hadamard is Avg&Dev all scaled up by a factor of $(\sqrt{2})^n$. Thus the first statement merely follows by the theorem.

(b) Hadamard is it's own undo, so just Hadamard both sides from part (a).

(c) This is definition of matrix-vector multiplication: we get the $|b_1 \dots b_n\rangle$ labeled column of A_n by multiplying A_n by $|b_1 \dots b_n\rangle$. But recall that A_n is just H_n divided by $(\sqrt{2})^n$ (Hadamard does $\sqrt{\frac{1}{2}}(x+y)$, Avg&Dev does $\frac{1}{2}(x+y)$). Thus, divide the expression from part (b) by $(\sqrt{2})^n$ on both sides to get $A_n |b_1 \dots b_n\rangle = (\sqrt{\frac{1}{2}})^n |\chi_b\rangle = (\sqrt{\frac{1}{2}})^n$ times $(\sqrt{\frac{1}{2}})^n$ times the ± 1 truth table of $XOR_{b_1 \dots b_n}$, which is $(\frac{1}{2})^n$ times the ± 1 truth table of $XOR_{b_1 \dots b_n}$.

(d) We get the $|b_1 \dots b_n\rangle$ labeled row of A_n by multiplying $|b_1 \dots b_n\rangle^T$ by A_n. But A_n like Hadamard H_n is symmetric (since Hadamard or Avg&Dev don't involve complex values, the unitary property of H_n implies Hermitian implies symmetric). So this is the same as $|b_1 \dots b_n\rangle^T A_n^T = |\chi_b\rangle^T$ by part (c).

(e) Start in an n-qubit state where the amplitudes are the ± 1 truth table values of f. Then we're in a state $|\chi_f\rangle$.

When we do $A_n |\chi_f\rangle$, we do a series of row-column multiplications (basically dot production between each row of A_n and $|\chi_f\rangle$). Each row of A_n is the truth table of $XOR_{b_1 \dots b_n}$ times $(\frac{1}{2})^n$

Let's look at the inner product of row i of A_n with $|\chi_f\rangle$. Also, let's assume i is the base 10 representation of the binary number encoded in the binary string $b_1 \dots b_n$, so we will also refer to $XOR_{b_1 \dots b_n}$ as XOR_i.

Then the resulting ith entry is $(\frac{1}{2})^n |\chi_i\rangle \cdot |\chi_f\rangle$. But $|\chi_i\rangle \cdot |\chi_f\rangle$ is the same as the sum of the number of agreements between f and XOR_i minus the number of disagreements (since $1(1) = (-1)(-1) = 1$, while $1(-1) = (-1)(1) - 1$). Then the ith entry of $A_n |\chi_f\rangle$ is the amplitude on $|b_1 \dots b_n\rangle$, which is

$$(\frac{1}{2})^n (\#\text{ of agreements between } XOR_i \text{ and } f - \#\text{ of disagreements between } XOR_i \text{ and } f)$$

which is the definition of $\text{corr}(f, XOR_i)$ (exercises for Video #033/100). Replacing our notation to match the problem wording, we've shown the amplitude on $|b_1 \dots b_n\rangle$ is equal to $\text{corr}(f, XOR_{b_1 \dots b_n})$, as desired.

35 Kets

Video #035/100 Table of Contents

Instead of "x amplitude on $|0\rangle$ y amplitude on $|1\rangle$", where $x^2 + y^2 = 1$ and $x, y \in \mathbb{R}$, we can just represent a state as a *unit* vector. $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ and $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$, so $\begin{bmatrix}x\\y\end{bmatrix} = x\,|0\rangle + y\,|1\rangle$.

For a physical qubit, like maybe a photon, we can think of this vector as showing the direction of the photon's polarization. It's nice to see the geometric-real world connection.

I've already been using this sort of vector notation, especially in the notes for earlier videos. It was even used in the answer for the previous exercise (converting base 10 equivalent for a bitstring is kind of the same way).

Dirac's "ket" notation: $|\rangle$ signifies "datatype = column vector" AKA "ket".

35.1 Exercises

Consider these expressions:

(a) $0.6\,|0\rangle - 0.8\,|1\rangle$

(b) $\begin{bmatrix}0.6\\-0.8\end{bmatrix}$

Which of them is a valid name for the state "amplitude 0.6 on $|0\rangle$, amplitude -0.8 on $|1\rangle$"? Choose all that apply.

What is another name for the 2-qubit state $\begin{bmatrix}0.36\\0\\-0.48\\0.8\end{bmatrix}$?

35.2 Solutions

- Both (a) and (b) are valid synonyms
- That 2-qubit state is equivalent to $0.36\,|00\rangle - 0.48\,|10\rangle + 0.8\,|11\rangle$.

36 Toggle and Hadamard are Reflections

Video #036/100 Table of Contents

"toggle A" makes $|0\rangle \to |1\rangle$ and $|1\rangle \to |0\rangle$. In general, toggle does $x|0\rangle + y|0\rangle \to y|0\rangle + x|0\rangle$. We're reflecting over the line $y = x$ or $\theta = 45\,\text{deg}$.

- Toggle is it's own undo - makes sense reflection wise.
- Toggle preserves sum of squares of amplitudes, like how reflection preserves length of a vector
- Thus reflections are unitary!

This also tells us that the states $(\sqrt{\frac{1}{2}}, \sqrt{\frac{1}{2}}) = |+\rangle$ and $(-\sqrt{\frac{1}{2}}, -\sqrt{\frac{1}{2}}) = -|+\rangle$ are not changed by toggle (included common $|+\rangle$ notation for them.). Also, note that $(\sqrt{\frac{1}{2}}, -\sqrt{\frac{1}{2}}) = |-\rangle$ and $(-\sqrt{\frac{1}{2}}, \sqrt{\frac{1}{2}}) = -|-\rangle$ are perfectly negated by toggle. Conveniently, $|+\rangle$ and $|-\rangle$ also create the Hadamard matrix H: using augmentation, $H = \Big[|+\rangle \quad |-\rangle\Big]$.

Now, for Hadamard, let's note that $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$. But that makes perfect sense, since $|0\rangle = (1, 0)$ which gets the first column of H, then $|1\rangle = (0, 1)$ gets the second column of H. Note that Hadamard is also a reflection across $\theta = 22.5\,\text{deg} = \frac{\pi}{16}\text{rad}$.

Note that Had is a linear transformation (proven by O'Donnell using amplitude trees, which I refuse to LaTeX) on $(x, y) = x|0\rangle + y|1\rangle$ gives $x|+\rangle + y|-\rangle = \sqrt{\frac{1}{2}}(x + y, x - y)$ (this shows linearity of Hadamard, so Had is an L.T.).

Any quantum operation is a linear transformation (their behavior follows the amplitude tree rules).

36.1 Exercises

As you recall, "toggle" corresponds to the following operation on $\mathbb{R}^2$: "reflection through the line at angle 45°". Is it possible to do a toggle "halfway"? Unfortunately, there's no linear transformation S on $\mathbb{R}^2$ such that $S \cdot S$ (i.e., doing S twice) is equivalent to toggle. But... in our real world, you can "reflect" a piece of paper by rotating it through the third dimension... and you could furthermore stop this rotation halfway.

Bearing this in mind, and calling the 3 standard basis vectors of $\mathbb{R}^3$ by the names $|0\rangle, |1\rangle, |2\rangle$, find a real unitary 3-dimensional transformation T with the property that $T \cdot T$ maps $|0\rangle \to |1\rangle, |1\rangle \to |0\rangle, |2\rangle \to \pm|2\rangle$. In addition to finally showing a matrix for T, give a complete narrative of how you worked it out.

36.2 Solutions TODO

Rotation through 3rd dimension. Create matrix A corresponding to L.T. (linear transformation) T. Let A_i be the ith col of A. We want $A^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & \pm 1 \end{bmatrix}$. The determinant of A^2 is ± 1 (we can do one row swap to get into a diagonal form. $\det A^2 = \mp 1 \iff A_{3,3}^2 = \pm 1$). But we need $\det A^2$ to be postive, since $\det A^2 = \det A \times \det A$, and $\det A$ will only be real if $\det A^2 \geq 0$. So we want $A_{3,3}^2 = -1$; $A^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$

Then $\det A^2 = 1$, which means $\det A$ is either 1 or -1.

Using a sticky note, we see that the only way to preserve orientation while moving the x-axis to the y-axis and vice versa is for the z-axis to become the -z-axis, so we're good so far.

I plugged things into a python symbolic solver, and it's failing (didn't yield an answer after 11 minutes). So what if the bottom right entry was actually positive 1?

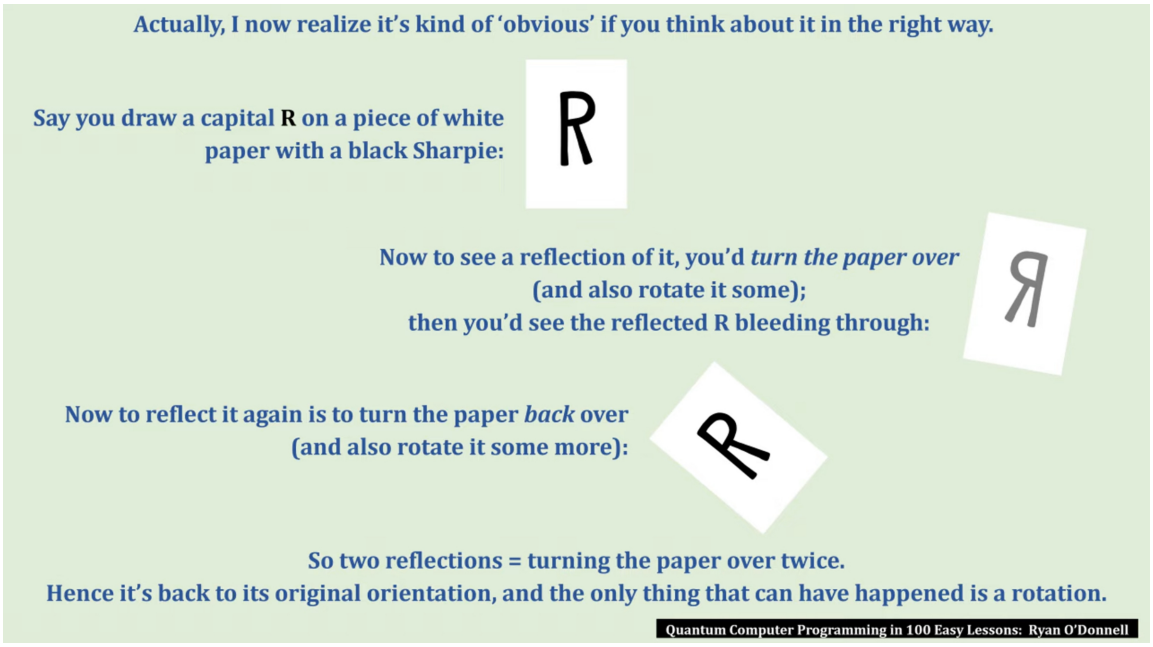
37 Rotations in 2-D

Video #037/100 Table of Contents

`NewOp(A)`

1. `Hadamard A`
2. `toggle A`

We've composed 2 reflections here. Geometrically, two reflections equals a rotation. If two reflection axes are at angle θ, then the composite operation is a rotation by 2θ. If a vector starts at angle α from the first axis, the first reflection rotates it 2α. The second rotates is $2(\theta - \alpha)$. Add them together, you get a rotation by 2θ overall.



The angle between Hadamard and Toggle is $45\,\text{deg} - 22.5\,\text{deg} = 22.5\,\text{deg}$. Double it and we get that NewOp is a rotation by $45\,\text{deg} = \frac{\pi}{8}\text{rad}$. Thus, 8 consecutive NewOps is equivalent to doing nothing.

Recall that `Clockwise` had path matrix $\begin{bmatrix} .8 & .6 \\ -.6 & .8 \end{bmatrix}$, and `Counterclockwise` had path matrix $\begin{bmatrix} .8 & -.6 \\ .6 & .8 \end{bmatrix}$. These are rotation matrices, as I already stated in the notes for Video #009/100.

Consider: `NewNewOp(A): repeat 10 times: counterclockwise A` - basically ends up being rotation by 8.7 deg. Funnily enough, CCW 41 times gets to about -0.1 turns which is a really small clockwise rotation. But for sake of simplicity, Professor O'Donnell grants us rotation by any angle we want. `Rotation by` θ on A for any angle θ (physically possible by laws of physics, so it's a quantum instruction!)

Fact: any (real) unitary linear transformation is a rotation or reflection. In any number of dimensions.

37.1 Exercises

For this problem, let `turn A` be a new qubit manipulation instruction that transforms $|0\rangle$ to $|1\rangle$ and transforms $|1\rangle$ to $-1\,|0\rangle + 0\,|1\rangle$.

(a) Suppose you are writing a qubit-manipulation subroutine operating on a single qubit A, and the only instruction you are allowed to use is "turn A". Let us say that two subroutines are the "same" if they have the same matrix representation. How many different subroutines is it possible to make?

(b) Same question, but now you are allowed to use two instructions,"turn A" and "toggle A".

(c) Same question, but now you are allowed to use two instructions,"turn A" and "Hadamard A".

(d) Same question, but now you are allowed to use the single instruction "clockwise A". (Hint: you may look up and appeal to "Niven's Theorem')

(e) Do your best to show that every physically possible 1-qubit transformation (only involving real amplitudes) can be arbitrarily-closely approximated by a subroutine that only uses the instructions "toggle A" and "clockwise A"

37.2 Solutions

(a) turn is actually just a 90 degree CCW rotation. So there's only 4 possible operations (1,2,3,4 turns, mod 4 cycle)

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(b) We've got the original 4 from turn, and we have toggle, which has path matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Thus we can swap the rows of any of the turn matrices, yielding $2 \times 4 = 8$ possible subroutines. Since toggle is it's own inverse, it wouldn't make sense to toggle twice.

(c) Hadamard A is also a reflection, just like toggle. Hence it's in own inverse, so the same logic applies as from part (b). 8 possible subroutines.

(d) Because the irrationals are dense, and Clockwise rotations by an irrational number of degrees, there's going to be an infinite number of angles you can hit with repeated clockwises (as mentioned in lecture, you can clockwise 41 times and get about 0.1 turns, and since each turn is irrational number of degrees, you'll keep going for infinitely many angles).

So now assume this single qubit A starts at $|0\rangle$. Then the state of the qubit is of the form (x, y) where x and y are rational. After applying clockwise, state vector (x, y) becomes Clockwise$(x, y) = (0.8x + 0.6y, -0.6x + 0.8y)$ which will be rational as rationals are closed under multiplication and addition. Thus the only rational values of θ we could hit are $0, 30, 90$ degrees. But there's infinitely many rationals anyway so Niven's theorem doesn't reduce the number of possible subroutines.

Niven's Theorem states that the only rational values of θ in the interval $0° \leq \theta \leq 90°$ for which $\sin\theta$ is also a rational number are $\sin 0° = 0$, $\sin 30° = \frac{1}{2}$, and $\sin 90° = 1$.

(e) As stated in lecture, we can rotate by any arbitrary θ using repeated clockwise. We need toggle to invert the orientation of space (since toggle has negative determinant). So we can literally do any 1-qubit operation (AKA any unitary linear transformation in $\mathbb{R}^2$) as long as we can invert space and reach any rotation.

38 Every Rotation is 2-D Rotations (Lecture 11)

Video #038/100 Table of Contents

Recall "unitary" = "doesn't change lengths"
$\implies$ doesn't change angles

Fix vector $|v\rangle$ and $|w\rangle$. AFSOC unitary transformation preserves length and does not preserve angle. Then the vector $|w\rangle - |v\rangle$ wouldn't have been transformed correctly. This is due to the linearity of the unitary transformations (which we represent as linear transformations). So then all unitary transformations are rotations or reflections.

Let's think of what happens in 3-D (not a possible space for qubits, but it works nonetheless).
Theorem (Euler): There are no "inherently 3-D rotations" - AKA, every 3-D rotation is a 2-D rotation in a 2-D plane. This generalizes to higher-dimensional rotations.

More generally, Suppose U is a (real) unitary transformation in d-dim. Then there are some perpendicular (orthogonal technically) 2-dim planes $P_1 \dots P_r$ $(r \geq 0)$ and associated angles $\theta_1 \dots \theta_r$ $\theta_i \neq 0\,\text{deg}, 180\,\text{deg}, \in [0, 360]\,\text{deg}$ such that U does "rotation by θ_j" in plane P_j. In each of the remaining $d - 2r$ dims, U either does negation $\theta = 180\,\text{deg}$ or nothing $\theta = 0\,\text{deg}$

In essence, higher dimensional rotations fix several orthogonal axes of rotation and perform some rotation about each axis. However, since each axis is a normal to a 2-D plane, we can think of it as a bunch of orthogonal planes within which rotation occurs.

38.1 Exercises

Suppose Q is a 3-D "rotation", by this we mean a real unitary linear transformation that "preserves orientation" (Here preserves orientation means that if you imagine U having been performed on a globe, then when you look at North America with Canada on top, you have Pennsylvania on the right and California on the left and not the other way around). The goal of this problem is to show that there must be some one-dimensional axis such that Q fixes this axis (and hence Q acts like a rotation by some angle $0\,\text{deg} \leq \theta_1 < 360\,\text{deg}$ in the two-dimensional plane perpendicular to the axis).

(a) Explain why we are done if $Q|x\rangle = |x\rangle$ for every point $|x\rangle$ on the unit sphere.

(b) Otherwise, let $|u\rangle$ be a point on the unit sphere with $Q|u\rangle \neq |u\rangle$. Write $|u'\rangle = Q|u\rangle$. Explain why there is a reflection R_1 through some 2-D plane (through the origin) such that $R_1|u'\rangle = |u\rangle$. Further explain why $R_1 Q|x\rangle \in V$ for all vectors in $|x\rangle \in V$, where V is the 2-D plane perpendicular to $|u\rangle$.

(c) Explain why there must be some unit vector $|v\rangle \in V$ such that $R_1 Q|v\rangle \neq |v\rangle$. Otherwise, argue that Q must not be preserving orientation.

(d) Choosing such a $|v\rangle$ and writing $|v'\rangle$ for $R_1 Q|v\rangle$, explain why there is a reflection R_2 through some 2-D plane containing $|u\rangle$ such that $R_2|v'\rangle = |v\rangle$.

(e) Argue that $R_2 R_1 Q|u\rangle = |u\rangle$, $R_2 R_1 Q|v\rangle = |v\rangle$, and $R_2 R_1 Q|w\rangle \in L$ for all $|w\rangle \in L$, where L is the line perpendicular to $|u\rangle$ and $|v\rangle$.

(f) Argue that $R_2 R_1 Q$ cannot be acting as negation within L (hint: orientation again). Deduce that Q is the product of two reflections, and hence a 2-D rotation about the line in common to the reflection planes of R_1 and R_2.

38.2 Solutions

(a) We WTS Q is a rotation by some angle about some 1-D axis. If Q is the identity transformation, i.e. $Q|x\rangle = |x\rangle$ for any $|x\rangle$ on the unit sphere, then we can pick any arbitrary $1-D$ axis (any line that passes through the origin) and perform rotation by 0 deg. So we're done.

(b) Choose the 2-D plane (through the origin) that includes the "midpoint" vector, whose entries are the average of $|u\rangle$ and $|u'\rangle$ and is perpendicular to the vector $|u\rangle - |u'\rangle$. Reflection through this plane brings $|u'\rangle$ back to $|u\rangle$. Denote this plane $\mathcal{P}_1$.

Now fix $|x\rangle \in V$, where V is the plane of all vectors $|x\rangle$ such that $|x\rangle \perp |u\rangle$. Then $Q|x\rangle \perp Q|u\rangle = |u'\rangle$. Then $R_1 Q|x\rangle \perp R_1|u'\rangle = |u\rangle$. Since $R_1 Q|x\rangle \perp |u\rangle$, then $R_1 Q|x\rangle \in V$.

(c) AFSOC $R_1 Q|v\rangle = |v\rangle$ for all $|v\rangle \in V$. Then Q would be R_1, since reflections are their own undo. Then Q would be a reflection, so Q would not preserve orientation (as reflections, by definition, reverse orientation).

(d) Choose the 2-D plane (through the origin) that includes the "midpoint" vector, whose entries are the average of $|v\rangle$ and $|v'\rangle$ and is perpendicular to the vector $|v\rangle - |v'\rangle$. Reflection through this plane brings $|v'\rangle$ back to $|v\rangle$. Denote this plane $\mathcal{P}_2$.

Note $\mathcal{P}_2$ must contain $|u\rangle$. $\mathcal{P}_2 \perp |v\rangle - |v'\rangle$, so $\mathcal{P}_2$ contains all vectors $|x\rangle$ such that $|x\rangle \cdot (|v\rangle - |v'\rangle) = 0$.

By linearity of dot product, this means $|x\rangle \cdot |v\rangle - |x\rangle \cdot |v'\rangle = 0$. Let $|x\rangle = |u\rangle$. Since $|v\rangle$ and $|v'\rangle$ are both in V (proven in part b), and $V \perp |u\rangle$, then $|x\rangle \cdot |v\rangle - |x\rangle \cdot |v'\rangle = 0 - 0 = 0$. So $|u\rangle$ is in $\mathcal{P}_2$.

(e) $R_2 R_1 Q|u\rangle = R_2|u\rangle$, But since $|u\rangle$ is already $\mathcal{P}_2$, it is unchanged by R_2. Hence $R_2|u\rangle = |u\rangle$.

$R_2 R_1 Q|v\rangle = R_2|v'\rangle = |v\rangle$ as we defined R_2 in part d.

Fix $|w\rangle \in L$. Then $|w\rangle \perp |u\rangle$ and $|w\rangle \perp |v\rangle$.
$\implies R_2 R_1 Q|w\rangle \perp R_2 R_1 Q|u\rangle$ and $R_2 R_1 Q|w\rangle \perp R_2 R_1 Q|v\rangle$ by linearity
$\implies R_2 R_1 Q|w\rangle \perp |u\rangle$ and $R_2 R_1 Q|w\rangle \perp |v\rangle$ (we just proved $R_2 R_1 Q|u\rangle = |u\rangle$, $R_2 R_1 Q|v\rangle = |v\rangle$)
$\implies R_2 R_1 Q|w\rangle \in L$, as desired.

(f) As proven in part c, Q preserves orientation. We know R_1 and R_2 are nontrivial reflections, so they each reverse orientation. Thus $R_2 R_1$ overall preserves orientation (can use sign of determinants argument, but no need - just argue reversing something twice brings it back to itself). Then the composite transformation $R_2 R_1 Q$ must also preserve orientation, which means it cannot act as negation (since negation by definition does not preserve orientation).

Since $R_2 R_1 Q|w\rangle \in L$, $R_2 R_1 Q$ must be unitary, and $R_2 R_1 Q|w\rangle \neq -|w\rangle$, we can safely deduce $R_2 R_1 Q|w\rangle = |w\rangle$ (we need to stay on L, so $R_2 R_1 Q|w\rangle$ is an L.C. of $|w\rangle$, but it's not a negation of $|w\rangle$ and it can't be longer or shorter than $|w\rangle$ because the transformations are unitary, so the result must be $|w\rangle$ itself).

Then Q is the product of two reflections, namely $Q = R_1 R_2$. Then $R_2 R_1 Q = R_2 R_1 R_1 R_2$. Since reflections undo themselves, this means $R_2 R_1 Q = R_2 R_1 R_1 R_2 = R_2 I R_2 = R_2 R_2 = I$.

Because Q is two reflections, it must be a rotation (fact from Video #037/100). We're done.

39 From North Pole to Gabon to Singapore

Video #039/100 Table of Contents

Showing this statement from the previous video: More generally, Suppose U is a (real) unitary transformation in d-dim. Then there are some perpendicular (orthogonal technically) 2-dim planes $P_1 \dots P_r$ $(r \geq 0)$ and associated angles $\theta_1 \dots \theta_r$ $\theta_i \neq 0\,\text{deg}, 180\,\text{deg}, \in [0, 360]\,\text{deg}$ such that U does "rotation by θ_j" in plane P_j. In each of the remaining $d - 2r$ dims, U either does negation $\theta = 180\,\text{deg}$ or nothing $\theta = 0\,\text{deg}$

Let's think about $d = 1$. Then any unitary transformation either does nothing or does negation.

$d = 2$. Case on the value of r.

- $r = 1$. Then $P_1 = \mathbb{R}^2$. So we just rotate by not zero or 180 degrees.
- $r = 0$. Then we can do nothing/negate in each of the $d - 2(0) = d = 2$ remaining dimensions.

  if we negate in one dim and do nothing in the other, same as reflecting across non-negated axis.

$d = 3$. Case on the value of r.

- $r = 1$. Then P_1 is some 2-dim plane. So we just rotate by not zero or 180 degrees. Then the remaining $d - 2(1) = 3 - 2 = 1$ dimension either has negation or nothing.
- $r = 0$. Then we can do nothing/negate in each of the $d - 2(0) = d = 3$ remaining dimensions.

Back to quantum instructions: IncrMod3 from Video #001/100! It actually works in 3-D:

- $|00\rangle \to |01\rangle$, $|01\rangle \to |10\rangle$, $|10\rangle \to |11\rangle$, $|11\rangle \to |11\rangle$. Then we get $(x, y, z, c) = (z, x, y, c)$, where c remains constant. Treat $(1, 0, 0)$ as North Pole, $(0, 1, 0)$ as Gabon, and $(0, 0, 1)$ as Singapore.

39.1 Exercises

same as Video #038/100

39.2 Solutions

same as Video #038/100

But now we should revisit IncrMod3, which I previously proved was impossible using classical reversible instructions. The matrix for IncrMod3 is actually $\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. This matrix is actually invertible, so now we will have fun trying to make it.

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ c \end{bmatrix} = \begin{bmatrix} z \\ x \\ y \\ c \end{bmatrix}$$

This actually reveals that my initial understanding of the problem was wrong. We instead want the following mapping: $|00\rangle \to |01\rangle$, $|01\rangle \to |10\rangle$, $|10\rangle \to |00\rangle$, $|11\rangle \to |11\rangle$. This is actually invertible, and thus solvable. After watching this video, I updated exercises solutions for Video #001/100.

40 Angles and Rotation Matrices

Video #040/100 Table of Contents

2-D Rotations

We're allowed to use the new 1-qubit instruction, Rot_θ on A. Its paths matrix is $\begin{bmatrix} x & -y \\ y & x \end{bmatrix}$ and its inverse is $\begin{bmatrix} x & y \\ -y & x \end{bmatrix}$. Note $(Rot_\theta)^{-1} = (Rot_\theta)^\dagger$. Replace $x = \cos\theta$ and $y = \sin\theta$.

If θ is small, $\sin\theta \approx \theta$. (helpful fact for physics and stuff, can use Taylor series ig)

If a qubit's state vector is at angle θ, i.e. state $\cos\theta\,|0\rangle + \sin\theta\,|1\rangle$ and you extract, the probability for $|1\rangle$ is $\sin^2\theta \approx \theta^2$ (important for stuff like Grover's Algorithm for SAT).

40.1 Exercises

40.1.1 Linear Algebra Fun.

Let $M \in \mathbb{R}^{d\times d}$ be a matrix. You should think of M as a linear transformation on R^d. The matrix M is called "normal" if $MM^\dagger = M^\dagger M$.

Tyler's note: this seems like an extension of orthogonal matrices from 21-241, extended for the complex numbers. Indeed, Wikipedia corroborates this fun tidbit because the Spectral Theorem (which in 21-241, stated that a symmetric matrix S is orthogonally diagonalizable into QDQ^T) generalized states that a martix is normal implies it is unitarily similar to a diagonal matrix and thus diagonalizable into $UDU^\dagger$. Should've read more into 21-242.

Every matrix you encounter in quantum computing is normal: "unitary", "orthogonal", "symmetric", "skew-symmetric", "Hermitian", "skew-Hermitian", "rotation", "reflection", "adjacency matrix of an undirected graph", "permutation matrix" — all of these classes of matrices are normal. In 2-D pictures, normal matrices/transformations are the ones that don't do any "shearing".

The awesomest kind of a normal matrix M is one that I'll call a "basis scaling". This just means that there is some orthonormal basis $|b_1\rangle, \ldots, |b_d\rangle$ (some alternative axes for $\mathbb{R}^d$, if you will) and some scalars $c_1 \ldots c_d$ sucht aht the action of M is to scale by c_1 in the $|b_1\rangle$ direction, ..., scale by c_d in the $|b_d\rangle$ direction. These kinds of M are so great because ... well ... how much simpler can you get? You're just stretching space by differing scale-factors in different directinos.

But wait, what about this rotation matrix on $\mathbb{R}^2$, $Rot_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ where $0 \le \theta < 2\pi$. It's normal, and it's pretty simple-looking, but...

40.1.2 The actual exercises, and the "E" Word

(a) Explain why geometrically why Rot_θ is not awesome — i.e., not a basis-scaling — except for the two cases

$$Rot_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad Rot_{\frac{\tau}{2}} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

where τ denotes 2π, a full turn of the circle.

(b) Let's focus on a moment on rotation by 90°.

$$Rot_{\frac{\tau}{4}} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

So we're pretty sad, because we wish $Rot_{\frac{\tau}{4}}$ were an awesome basis-scaling but it's not. Or is it? Suppose we were really hoping that $Rot_{\frac{\tau}{4}} \begin{bmatrix} x \\ y \end{bmatrix} = c \begin{bmatrix} x \\ y \end{bmatrix}$ is true for some scalar c (and excluding the case $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$). Write down the two equations this would give. Multiply the two equations together and divide out any common factor (if doing this makes you squeamish, you can quickly show that neither x nor y can be 0 or else both would be). This leaves one famous equation involving c. Does the history of math prompt us to declare that it has a solution? (In fact, two solutions...?)

(c) Identify two orthonormal vectors $|b_1\rangle, |b_2\rangle$ (meaning $\langle b_1|b_1\rangle = \langle b_2|b_2\rangle = 1$ and $\langle b_1|b_2\rangle = 0$), each of which is "scaled" by one of the two solutions for c from the previous problem.

(d) Show that in fact for any Rot_θ, there are "scalars" c_1 and c_2 such that $Rot_\theta |b_1\rangle = c_1 |b_1\rangle$ and $Rot_\theta |b_2\rangle = c_2 |c_2\rangle$.

Remark: once you go down this road, you can show that indeed every normal $d \times d$ matrix M is in fact basis-scaling. How awesome! But we will not go down this road in 15-459, which is why we can only break real unitary M's down as far as 2×2 rotation matrices.

40.2 Solutions

(a) Rot_0 is a basis scaling that scales the $|0\rangle$ and $|1\rangle$ axes by 1; $Rot_{\frac{\tau}{2}}$ scales the $|0\rangle$ and $|1\rangle$ axes by -1.

(b) Want $\begin{bmatrix} -y \\ x \end{bmatrix} = c \begin{bmatrix} x \\ y \end{bmatrix}$. Then $-y = cx$ and $x = cy$.

Of course, if either of x or y were zero, then we'd get both x and y are zero. ($0 = cx \wedge x = 0$ or $-y = 0 \wedge 0 = cy$).

Now we multiply to get $-xy = c^2xy \implies -1 = c^2$. I don't really know if this is "famous", but this does tell us that $c = \pm i$ — two solutions!

(c) What vectors satisfy the property? Using $c = i$, $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -y \\ x \end{bmatrix} = i \begin{bmatrix} x \\ y \end{bmatrix} \implies \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ -ix \end{bmatrix}$

Now using $c = -i$, $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -y \\ x \end{bmatrix} = -i \begin{bmatrix} x \\ y \end{bmatrix} \implies \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ ix \end{bmatrix}$

Now we need to show these vectors are orthonormal.

$(x, -ix)^\dagger(x, -ix) = \overline{(x, -ix)} \cdot (x, -ix) = (x, ix) \cdot (x, -ix) = x^2 + (-i^2x^2) = 2x^2$

$(x, ix)^\dagger(x, ix) = \overline{(x, ix)} \cdot (x, ix) = (x, -ix) \cdot (x, ix) = x^2 + (-i^2x^2) = 2x^2$

$(x, ix)^\dagger(x, -ix) = \overline{(x, ix)} \cdot (x, -ix) = (x, -ix) \cdot (x, -ix) = x^2 + (i^2x^2) = 0.$

So just set $x = \sqrt{\frac{1}{2}}$.

(d) Recall $|b1\rangle = (x, -ix)$ and $|b_2\rangle = (x, ix)$.

$Rot_\theta |b1\rangle = \begin{bmatrix} x\cos\theta + ix\sin\theta \\ x\sin\theta - ix\cos\theta \end{bmatrix} = \begin{bmatrix} cx \\ -icx \end{bmatrix}$

$\implies -ic^2x^2 = x^2\sin\theta\cos\theta - ix^2\cos^2\theta + ix^2\sin^2\theta - i^2x^2\sin\theta\cos\theta$

$\implies -ic^2x^2 = x^2\sin\theta\cos\theta - ix^2(\cos^2\theta - \sin^2\theta) + x^2\sin\theta\cos\theta$

$\implies -ic^2x^2 = x^2(2\sin\theta\cos\theta) - ix^2(\cos^2\theta - \sin^2\theta)$

$\implies -ic^2x^2 = x^2(\sin 2\theta) - ix^2(\cos 2\theta)$

$\implies -ic^2 = (\sin 2\theta) - i(\cos 2\theta)$

$\implies c^2 = \cos 2\theta + i\sin 2\theta$

$\implies c^2 = cis(2\theta) \implies c = cis(\theta)$ (De Moivre's Theorem)

$Rot_\theta |b2\rangle = \begin{bmatrix} x\cos\theta - ix\sin\theta \\ x\sin\theta + ix\cos\theta \end{bmatrix} = \begin{bmatrix} cx \\ icx \end{bmatrix}$

$\implies ic^2x^2 = x^2\sin\theta\cos\theta + ix^2\cos^2\theta - ix^2\sin^2\theta - i^2x^2\sin\theta\cos\theta$

$\implies ic^2x^2 = x^2\sin\theta\cos\theta + ix^2(\cos^2\theta - \sin^2\theta) + x^2\sin\theta\cos\theta$

$\implies ic^2x^2 = x^2(2\sin\theta\cos\theta) + ix^2(\cos^2\theta - \sin^2\theta)$

$\implies ic^2x^2 = x^2(\sin 2\theta) + ix^2(\cos 2\theta)$

$\implies ic^2 = (\sin 2\theta) + i(\cos 2\theta)$

$\implies c^2 = \cos 2\theta - i\sin 2\theta = \cos -2\theta + i\sin -2\theta$

$\implies c^2 = cis(-2\theta) \implies c = cis(-\theta)$ (De Moivre's Theorem)

All of the implications are actually biimplications here because they're trig identities. Thus we've proven the desired result.

41 Dot Products

Video #041/100 Table of Contents

Say $|v\rangle = \begin{bmatrix} x \\ y \end{bmatrix} = x\,|0\rangle + y\,|1\rangle$.

We can recover x through the "dot product" of $|0\rangle$ and $|v\rangle = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = x(1) + y(0) = x$

The second half of Dirac's notation: $\begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^\dagger = \langle 0| = |0\rangle^\dagger$. These row vectors are called "bra"s, so we have "bra-ket" notation.

Now create $|f\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$. Then the "dot product" of $|f\rangle$, $|v\rangle$ is $\langle f|v\rangle = \langle f|v\rangle$ (the LaTeXfor these two is different, but they render the same way.)

For example, if $|w\rangle = (a, b, c, d)$ is a 2-qubit state, $\langle 11|w\rangle = d$. Then $\mathbb{P}(\text{`see } 11') = \langle 11|w\rangle^2 = d^2$

Why dot product? Well let's say $|f\rangle = Rot_\alpha\,|0\rangle$. Then $\langle f| = |f\rangle^\dagger = (Rot_\alpha\,|0\rangle)^\dagger = |0\rangle^\dagger\,Rot_\alpha^\dagger = |0\rangle\,Rot_{-\alpha}$ by defn of tranposing matrix product and using that the transpose of rotation is the same as rotating by the negative angle (recall how counterclockwise and clockwise were inverses?)

Then $\langle f|v\rangle = \langle f| \cdot |v\rangle = \langle 0|\,Rot_\alpha\,|v\rangle =$ the 0th coordinaate of $|v\rangle$ rotated by α.

Summary: assuming $|f\rangle$, $|v\rangle$, and $|g\rangle$ are all unit vectors,

$\langle f|v\rangle$ is $|v\rangle$'s coordinates on $|f\rangle$ in the $|f\rangle, |g\rangle$ basis (where $|g\rangle = \begin{bmatrix} -b \\ a \end{bmatrix}$, hence the matrix $\begin{bmatrix} |f\rangle & |g\rangle \end{bmatrix} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix} = Rot_\alpha$),
or the length of $|v\rangle$'s projection onto f,
or $\cos\gamma$, where γ is the angle between $|v\rangle$ and $|f\rangle$.

Similarly, $\langle g|v\rangle$ =length of $|v\rangle$'s project onto $|g\rangle = \cos\phi$, where ϕ is the angle between $|v\rangle$ and $|g\rangle$. Note that by definition of rotation matrix, $|f\rangle$ and $|g\rangle$ are at angle $\frac{\pi}{2}$ to each other, so $\gamma + \phi = \frac{\pi}{2}$.

41.1 Exercises

Linear algebra For all "scalars" that appear in this problem (particularly, all matrix entries), I do not mind if you assume they are nonnegative integers. In fact, I strongly recommend it.

(a) Let C be an $m \times n$ matrix, let B be an $n \times p$ matrix, and let A be a $p \times q$ matrix. Prove that $(CB)A = C(BA)$, which means we can simply write CBA. (hint, you may take the Lesson 9 exercise for granted, in which case you will not need to write any symbols, merely words.)

(b) Let $\underline{A}$ be another $p \times q$ matrix. Prove that $B(A + \underline{A}) = BA + B\underline{A}$. Of course, we similarly have $(B + \underline{B})A = BA + \underline{B}A$, but you don't need to prove this. (hint, again, via the lesson 9 exercise, this can be proven using merely words. For example, "apricot" is basically the same color as "amber".)

(c) Let c be a scalar. Prove that $c(BA) = (cB)A = B(cA)$.

(d) Cf. Exercise 17, prove that $(A + \underline{A})^\dagger = A^\dagger + \underline{A}^\dagger$. Also prove $(cA)^\dagger = cA^\dagger$.

(e) I know we stated this in class, but prove that $(BA)^\dagger = A^\dagger B^\dagger$. (You can cite anything from HW1.3, 2.3, 3.1). Deduce that $(CBA)^\dagger = A^\dagger B^\dagger C^\dagger$ (Math nerds: if one is using complex numbers, then actually $(cA)^\dagger = c^\dagger A^\dagger$, where $c^\dagger$ denotes the complex conjugate of c.)

(f) If $|x\rangle$ is a height-p vector and $|y\rangle$ is a height-q vector, explain why $(\langle x| A |y\rangle)^\dagger = \langle y| A^\dagger |x\rangle$.

(g) Let $|a\rangle, |b\rangle, |g\rangle, |h\rangle$ be vectors of the same height and let $\alpha, \beta, \gamma, \delta$ be scalars. Write $|u\rangle = \alpha |a\rangle + \beta |b\rangle$ and write $|v\rangle = \gamma |g\rangle + \delta |h\rangle$. By referring to previous parts of this problem, annotate the justification for each step in the following computation of the dot product of $|u\rangle$ and $|v\rangle$.

$$\langle u|v\rangle = (\alpha |a\rangle + \beta |b\rangle)^\dagger (\gamma |g\rangle + \delta |h\rangle) \tag{1}$$
$$= (\alpha \langle a| + \beta \langle b|)(\gamma |g\rangle + \delta |h\rangle) \tag{2}$$
$$= \alpha\gamma \langle a|g\rangle + \alpha\delta \langle a|h\rangle + \beta\gamma \langle b|g\rangle + \beta\delta \langle b|h\rangle \tag{3}$$

(Math nerds: figure out what/where the complex-conjugated scalars ought to be in case we are using complex numbers).

(h) In the previous problem, suppose we are in two dimensions and $|a\rangle = |g\rangle = |0\rangle$ and $|b\rangle = |h\rangle = |1\rangle$. Explain how/why the formula above reduces to the "usual" dot product formula for 2-D vectors in terms of their entries.

41.2 Solutions

(a) Like from exercises for Video #009/100, we can form a $n \times q$ matrix $D = BA$ by gluing the $n \times p$ paths diagram B represents on top of the $p \times q$ paths diagram A represents. Then we can form a matrix $E = CD$ by gluing the $m \times n$ paths diagram C on top of the $n \times q$ paths diagram D represents. Then E is a $m \times q$ paths diagram. Similarly, we could've instead first glued C on top of B, creating a $m \times p$ paths diagram that we then glue on top of A, creating a $m \times q$ paths diagram. It doesn't matter what order we "glue" these paths diagrams together.

(b) We can first put the two $p \times q$ paths diagrams A and $\underline{A}$ next to each other, then glue B to each of them. Or we can glue B directly to the top of each one.

(c) We can just scale paths; we could scale B first then glue A to the bottom, scale A first then glue B on top, or glue A and B together then scale the whole thing.

(d) From exercises for Video #017/100, we know that † is just the conjugate transpose, which has linearity over addition. Linearity also applies for scalar multiplication, though the "math nerds" note is important if c is in complex and has a nontrivial imaginary component.

(e) No clue what those HW numbers represent (something to do with lecture number I presume, but the homework numbers are often whited-out for the exercises.) No matter. We'll just jump to the three matrix example: $(CBA)^\dagger = \overline{(CBA)^T} = \overline{A^T B^T C^T} = A^\dagger B^\dagger C^\dagger$.

(f) just use part (e) of these exercises. It just follows.

(g) (1) substitute the expansions of $|u\rangle$ and $|v\rangle$, and use the definition that $\langle u| = |u\rangle^\dagger$.
(2) use the definition that $\langle u| = |u\rangle^\dagger$, and result from part (d). Here, we would also put $\alpha^\dagger$ and $\beta^\dagger$ if the scalars were complex with nontrivial imaginary components.
(3) FOIL. Final answer completely generalized to complex numbers should be $\alpha^\dagger\gamma \langle a|g\rangle + \alpha^\dagger\delta \langle a|h\rangle + \beta^\dagger\gamma \langle b|g\rangle + \beta^\dagger\delta \langle b|h\rangle$

42 Measuring in a different basis

Video #042/100 Table of Contents

Suppose your algorithm has basis $\{|f\rangle, |g\rangle\}$ "in mind"; it knows α.

Lets create an algorithm to measure/extract in the $\{f, g\}$ basis.

ExtractIn$\{f, g\}$Basis(A):

1. $Rot_{-\alpha}$ on A
2. extract A, but

   if display 0, then display f instead

   if display 1, then display g instead

If the state of qubit A is $ketv$, what's $\mathbb{P}$("display f'')?

After line 1: state is $Rot_{-\alpha}|v\rangle$. So $\mathbb{P}(\text{"}f''\text{)} = \langle 0| Rot_{-\alpha}|v\rangle = \langle f|v\rangle^2 = \cos^2\gamma$

We've had this extract operation for awhile. It is destructive - the qubit is destroyed after being measured. But for some physical implementations of qubits, the physical object is still around, but the state "collapses" into the measured state. This is "non-destructive measurement", where the qubit is sitll there, but its new state has all the amplitude on whatever the displayed state was.

We'll call non-destructive measurement the "measure" command, but it's the same as

1. extract A — A is destroyed
2. a = output bit — get displayed state
3. new qubit A
4. if a then toggle A

Using this, we can create MeasureIn$\{f, g\}$Basis(A):

1. $Rot_{-\alpha}$ on A
2. extract A
3. a = output bit
4. new qubit A
5. if $a = 0$ then $Rot_{\alpha}(A)$
   else $a = 1$ then $Rot_{-\alpha}(A)$

3-D glasses and their lenses implement semi-destructive measurement. When a photon comes in, it sometimes destroys the photon and sometimes doesn't.

42.1 Exercises

- A qubit in the state $Rot_{\frac{\pi}{1000}}|0\rangle$ is measured in the standard $\{|0\rangle, |1\rangle\}$ basis. Bearing in mind that $\pi^2 \approx 10$, what is closest to the probability that the outcome is "1"?
- A qubit in the state $.6|0\rangle + .8|1\rangle$ is measured in the basis $\{|g\rangle, |h\rangle\}$ where $|g\rangle = .8|0\rangle + .6|1\rangle$ and $|h\rangle = -.6|0\rangle + .8|1\rangle$. What is the probability the outcome is g?

- Consider the following code:

  1. new qubit A
  2. Hadamard A
  3. Rotate 15° on A
  4. toggle A

  Now what is the angle that A's state vector makes with the horizontal axis (i.e., $|0\rangle$-axis)?

- Consider the following code:

  1. new qubit A
  2. Hadamard A
  3. Rotate 15° on A
  4. extract all

  What is the probability this prints "1"?

42.2 Solutions

- For small values of θ, $\langle 1| Rot_\theta |0\rangle \approx \theta$. Then $\mathbb{P}(\text{“}1''\text{)} = (\frac{\pi}{1000})^2 = \frac{\pi^2}{10^6} \approx \frac{10}{10^6} = \frac{1}{10000}$.
- We use the MeasureIn$\{g,h\}$Basis(A) subroutine. $Rot_{-\alpha}$ on $(.6,.8)$ is the same as $\begin{bmatrix} .8 & .6 \\ -.6 & .8 \end{bmatrix} \begin{bmatrix} .6 \\ .8 \end{bmatrix} = \begin{bmatrix} .96 \\ .28 \end{bmatrix}$. Then $\mathbb{P}(\text{“}g''\text{)} = .96^2 = .9216$.
- new qubit starts A at angle 0°. Hadamard reflects across the line $\theta = 22.5°$, which puts A at angle 45°. Rotate 15° brings A to 60°. Toggle A reflects across the line $\theta = 45°$, bringing A to 30° with the horizontal axis.
- As we computed in the previous part (but throwing out the toggle step), A is it angle 60° with the horizontal. Then A is in state $\cos 60° |0\rangle + \sin 60° |1\rangle$, so $\mathbb{P}(\text{“}1''\text{)} = \sin^2 60° = (\frac{\sqrt{3}}{2})^2 = \frac{3}{4}$

43 Fun with filters (Lecture 12)

Video #043/100 Table of Contents

If $|v\rangle$ is extracted in orthonormal basis $\{|g\rangle, |h\rangle\}$, output:
$\mathbb{P}(\text{“}g''\text{)} = \langle g|v\rangle^2 = \cos^2\theta$
$\mathbb{P}(\text{“}h''\text{)} = \langle h|v\rangle^2 = \sin^2\theta = \theta^2$ if θ is small

Non-destructive measure operation: Extracting, then the state of the qubit collapses (or we remake the qubit) to the measured outcome.

Polarizing Filter: performs semi-destructive measurement on photons. $|g\rangle = |0\rangle, |h\rangle = |1\rangle$. For a horizontally polarizing filter (HPF) (where $|g\rangle$ is the horizontal axis and $|h\rangle$ is the vertical axis),
photons with state $|0\rangle$ pass through. Hence they are not destroyed
photons with state $|1\rangle$ are blocked and heat up the filter. Hence they are destroyed.
A photon with state $\cos\theta\,|0\rangle + \sin\theta\,|1\rangle$ has a $\cos^2\theta$ probability of passing through (and thus a $\sin^2\theta$ probability of getting blocked).

43.0.1 Experiment 1:

Pass “random” light through the horizontal polarizing filter.

1. new qubit A
2. $\theta = \text{UnifRandom}(0, 0, 1.0) \times \tau$
3. Rotate θ on A

We claim that the probability A passes through is 50%. Intuition - states “pair” up. After line 2, the likelihood of getting a state that has x% chance of passing through is the same as the likelihood of getting a state that has $100 - x$% chance of passing through.

After passing through the HPF, all remaining photos have state $|0\rangle$.

43.0.2 Experiment 2:

Put a “VPF”, where $|g\rangle = |1\rangle, |h\rangle = |0\rangle$ after the HPF. Then 0% of our photons will pass all the way through.

43.0.3 Experiment 3:

Place a third filter in between the HPF and VPF, rotated by 45°, where $|g\rangle = |+\rangle, |h\rangle = |-\rangle$.

What fraction of the light makes it all the way through?

- After the HPF, all photons are in state $|0\rangle$.
- After the second filter, photons pass through with probability $\mathbb{P}(|+\rangle) = \langle +|0\rangle^2 = \frac{1}{2} = 50\%$
- After the VPF, photons pass through with probability $\mathbb{P}(|1\rangle) = \langle 1|+\rangle^2 = \frac{1}{4} = 25\%$

43.1 Exercises: Rotation from filters

Suppose you are implementing one qubit using the polarization of a single photon. You have just implemented "new qubit A", so your photon is now in the state $|0\rangle$. you would like your qubit A to instead be $Rot_\theta |0\rangle = \cos\theta |0\rangle + \sin\theta |1\rangle$ where θ is a certain angle you have in mind, $-\frac{\tau}{2} < \theta \leq \frac{\tau}{2}$ (where $\tau = 2\pi = 360°$).

Suppose you own n polarizing filters. Explain (with proof) how you can use them to build a physical gadget such that when A is passed through your gadget, it comes out (i.e., is not blocked / converted to heat) with very high probability, and when it *does* come out, it is for sure in state $Rot_\theta |0\rangle$. In particular, by "very high probability", we mean that the probability it *fails* to come out is at most $\frac{\theta^2}{n} \leq \frac{10}{n}$. You may use without proof that $|\sin x| \leq x$ for all x.

Tyler's Note: $\frac{\theta^2}{n} \leq \frac{10}{n}$ is derived from our max rotation having $|\pi|$ radians, and $\pi^2 \approx 10$ and $\pi^2 < 10$, as stated in the exercises for Video #042/100.

43.2 Solutions

Place each filter sequentially such that the next filter is rotated $\frac{\theta}{n}$ more than the previous filter. Then for $0 \leq i \leq n$, filter i has basis $|g\rangle = (\cos\frac{i\theta}{n}, \sin\frac{i\theta}{n}), |h\rangle = (-\sin\frac{\theta}{n}, \cos\frac{\theta}{n})$ (filter 0 sets up "new qubit A" and initializes photon state to $|0\rangle$).

43.2.1 Desired Output State

Proof. Filter n has $|g_n\rangle = (\cos\frac{n\theta}{n}, \sin\frac{n\theta}{n}) = (\cos\theta, \sin\theta), |h_2\rangle = (-\sin\frac{n\theta}{n}, \cos\frac{n\theta}{n}) = (-\sin\theta, \cos\theta)$. Thus, any photons that pass through all n filters will be in the state $\cos\theta |0\rangle + \sin\theta |1\rangle$, as desired. □

43.2.2 Probability of Passing Through Filter i

Proof. (by induction) Prove $p(n)$ for $n \in \mathbb{N}$, where

$$p(n) := \text{"at filter } 1 \leq i \leq n\text{, photons fail to come out with probability } \sin^2\left(\frac{\theta}{n}\right)"$$

Base Case: Filter 1 has $|g_1\rangle = (\cos\frac{\theta}{n}, \sin\frac{\theta}{n}), |h_1\rangle = (-\sin\frac{\theta}{n}, \cos\frac{\theta}{n})$. Photons go into Filter 1 at state $|0\rangle$.

$\implies$ photons pass through with $\mathbb{P}(|g_1\rangle) = \langle g_1|0\rangle^2 = \cos^2\left(\frac{\theta}{n}\right)$

$\implies$ photons fail to come out $\mathbb{P}(|h_1\rangle) = 1 - \mathbb{P}(|g_1\rangle) = \sin^2\left(\frac{\theta}{n}\right)$

Induction Step: Assume $p(i)$ for $1 \leq i \leq n$. WTS $p(i{+}1)$. Filter $i{+}1$ has $|g_{i+1}\rangle = \left(\cos\frac{(i+1)\theta}{n}, \sin\frac{(i+1)\theta}{n}\right), |h_{i+1}\rangle = \left(-\sin\frac{(i+1)\theta}{n}, \cos\frac{(i+1)\theta}{n}\right)$, and photons go into Filter $i+1$ at state $|g_i\rangle = \left(\cos\frac{(i)\theta}{n}, \sin\frac{(i)\theta}{n}\right)$.

$$\mathbb{P}(\text{pass}) = \mathbb{P}(|g_{i+1}\rangle) = \langle g_i + 1|g_i\rangle^2 = \left(\cos\frac{(i+1)\theta}{n}\cos\frac{i\theta}{n} + \sin\frac{(i+1)\theta}{n}\sin\frac{i\theta}{n}\right)^2 \qquad \text{(inner product)}$$

$$= \left(\cos\left(\frac{(i+1)\theta}{n} - \frac{i\theta}{n}\right)\right)^2 = \left(\cos\left(\frac{\theta}{n}\right)\right)^2 = \sin^2\left(\frac{\theta}{n}\right) \qquad \text{(cosine difference identity)}$$

$$\mathbb{P}(\text{block}) = \mathbb{P}(|h_{i+1}\rangle) = 1 - \mathbb{P}(|g_{i+1}\rangle) = 1 - \cos^2\left(\frac{\theta}{n}\right) = \sin^2\left(\frac{\theta}{n}\right) \qquad \text{(Pythagorean identity)}$$

By SPMI, we're done. □

43.2.3 How many filters do we really need?

WTS $\mathbb{P}(\text{“fails to come out''}) \leq \frac{\theta^2}{n} \leq \frac{10}{n}$. This comes from $|\theta| \leq \pi$.

In the next subsubsection, we will use Bernoulli's inequality, $(1+x)^n \geq 1+nx$ for $x > -1$ and $n \geq 1$. In this case, our x is $-\frac{\theta^2}{n^2}$, so we want $\frac{\theta^2}{n^2} < 1 \iff \theta^2 < n^2 \iff n > \pi \iff n \geq 4$. This means we need at least 4 filters to guarantee a very high probability of passing through all filters for any valid θ and n.

43.2.4 Very High Probability of Passing Through All Filters

WTS $\mathbb{P}(\text{“fails to come out''}) = 1 - \mathbb{P}(\text{“pass all } n \text{ filters''}) = 1 - \left(\cos^2\left(\frac{\theta}{n}\right)\right)^n$

Proof.

$$
\begin{aligned}
& |\sin x| \leq x && \text{(given identity)} \\
\implies & \left|\sin\frac{\theta}{n}\right| \leq \frac{\theta}{n} \\
\implies & \sin^2\frac{\theta}{n} \leq \frac{\theta^2}{n^2} \\
\implies & 1 - \sin^2\frac{\theta}{n} \geq 1 - \frac{\theta^2}{n^2} \\
\implies & \cos^2\frac{\theta}{n} \geq 1 - \frac{\theta^2}{n^2} && \text{(Pythagorean identity)} \\
\implies & \left(\cos^2\frac{\theta}{n}\right)^n \geq \left(1 - \frac{\theta^2}{n^2}\right)^n \\
\implies & \left(\cos^2\frac{\theta}{n}\right)^n \geq 1 - n\frac{\theta^2}{n^2} && \text{(Bernoulli's inequality, } n \geq 4\text{)} \\
\implies & \left(\cos^2\frac{\theta}{n}\right)^n \geq 1 - \frac{\theta^2}{n} \\
\implies & 1 - \left(\cos^2\frac{\theta}{n}\right)^n \leq \frac{\theta^2}{n} \\
\implies & \mathbb{P}(\text{“fails to come out''}) \leq \frac{\theta^2}{n} \leq \frac{10}{n}\text{, as desired.} && \square
\end{aligned}
$$

44 Discriminating 2 qubits, no false positives

Video #044/100 Table of Contents

Quantum state tomography, quantum state estimation: given a qubit or a bunch of copies of a qubit in the same state, what state is it in? This is really a hypothesis testing problem (statistics!)

Mom: promises that qubit A's state $|test\rangle$ is either $|0\rangle$ or $|1\rangle$.

Q: Is test $|0\rangle$? Yes or no.

A: Extract A. If "1" output yes, "0" output no. $\mathbb{P}(\text{correct}) = 100\%$

Dad: promises that qubit A's state $|test\rangle$ is either $|+\rangle$ or $|-\rangle$.

Q: Is test $|+\rangle$? Yes or no.

A: Extract in basis $\{|+\rangle, |-\rangle\}$, + yes - no (or just hadamard then extract A, 1 yes 0 no). $\mathbb{P}(\text{correct}) = 100\%$

But your Mom and Dad are trustworthy. Professor Ada is also trustworthy, but makes things interesting.

Prof Ada: Promises that qubit A's state is either $|0\rangle$ or $Rot_{60^\circ}|0\rangle$.

Q: Is test $Rot_{60^\circ}|0\rangle$? Yes or no.

A: Here, we just extract in standard basis. If we see "1", we should output "yes".

if we see "0", we output "no", but there's a 25% chance that $Rot_{60^\circ}|0\rangle$ will be measured as 0.

So we have a little confusion matrix, where a positive result is "test is $Rot_{60^\circ}|0\rangle$". This represents a One-sided error algorithm, AKA "yes-means-yes" algorithm.

TP: $\mathbb{P}(\text{answer yes given } |test\rangle = Rot_{60^\circ}|0\rangle)$ is 75%; Given $|test\rangle = Rot_{60^\circ}|0\rangle$, we answer yes AKA see 1 three-fourths of the time

TN: $\mathbb{P}(\text{answer no given } |test\rangle = |0\rangle)$ is 100%; Given $|test\rangle = |0\rangle$, we always answer no AKA see 0

FP: $\mathbb{P}(\text{answer yes given } |test\rangle = |0\rangle)$ is 0%; Given $|test\rangle = |0\rangle$, we never answer yes AKA see 1

FN: $\mathbb{P}(\text{answer no given } |test\rangle = Rot_{60^\circ}|0\rangle)$ is 25%; Given $|test\rangle = Rot_{60^\circ}|0\rangle$, we answer no AKA see 0 one-fourth of the time

44.1 Exercises

Like in Lecture 12, your mom gives you a qubit in state $|test\rangle$ which is either $|0\rangle$ or $|1\rangle$. Unlike in Lecture 12, let us assume that your mom chooses between the two possibilities by secretly flipping a fair (50-50) coin.

Suppose you do any unitary operation you want, and then you measure (in the standard basis). Prove that the "overall probability" you see "0" is exactly $\frac{1}{2}$. Here by "overall probability", I mean with respect to *both* your mom's coin flip and the randomness inherent in measuring.

44.2 Solutions

We can think of any unitary operation as a rotation by some amount θ. if heads, $A = |0\rangle$, then the rotated state is $(\cos\theta, \sin\theta)$. if tails, $A = |1\rangle$, then the rotated state is $(-\sin\theta, \cos\theta)$. Overall, $\mathbb{P}(\text{see } 0) = \mathbb{P}(\text{see } 0 \wedge A = |0\rangle) + \mathbb{P}(\text{see } 0 \wedge A = |1\rangle) = \cos^2\theta \cdot \frac{1}{2} + \sin^2\theta \cdot \frac{1}{2} = \frac{1}{2}\left(\cos^2\theta + \sin^2\theta\right) = \frac{1}{2}$.

45 Discriminating 2 qubits, 2-sided error

Video #045/100 Table of Contents

45.0.1 One-sided error, ctd.

For one-sided error algorithms, we can either guarantee no false positives or no false negatives, depending on the hypothesis. If we had asked the question "Is test $|0\rangle$? Yes or no" to Prof. Ada's scenario in Video #044/100, we would have guaranteed no false negatives instead of no false positives. Alternatively, we could've just measured in basis $\{Rot_{-30^\circ}|1\rangle, Rot_{60^\circ}|0\rangle\}$, corresponding to (no, yes) respectively.

The example in lecture is a "worst-case" analysis that means the overall probability of error is $\frac{1}{4}$, since we have 25% chance of FN given the desired result. However, if we were to analyze the exercise for Video #044/100, the probability of error in the exercise would end up being $\frac{1}{2} \times \frac{1}{4} = \frac{1}{8}$, since there's a 50% chance of the desired result, then a 25% chance of falsely rejecting the hypothesis given the desired result.

45.0.2 Two-sided error

Recall Prof Ada Promises that qubit A's state is either $|0\rangle$ or $Rot_{60^\circ}|0\rangle$.
H : Is test $Rot_{60^\circ}|0\rangle$? Yes or no. To solve, measure in an orthonormal basis that is "symmetric" with the potential states. Measure with $|yes\rangle = Rot_{75^\circ}|0\rangle, |no\rangle = Rot_{-15^\circ}|0\rangle$. In both cases, $|test\rangle$ is 15° away from the correct basis vector. In either case, $A = |0\rangle$ or $A = Rot_{60^\circ}|0\rangle$, Then $\mathbb{P}(\checkmark) = \cos^2 15^\circ$ and $\mathbb{P}(\times) = \sin^2 15^\circ$. Both vectors are 15° away from the right answer and 75° away from the wrong answer.

45.1 Exercises

Same question as before (Video #044/100), except with your dad, who secretly flips a fair coin to decide whether $|test\rangle$ is $|+\rangle$ or $|-\rangle$.

45.2 Solutions

We can think of any unitary operation as a rotation by some amount θ. We can just bring $|+\rangle, |-\rangle$ back to $|0\rangle, |1\rangle$ by doing Hadamard A. Then it's the same exercise with the same result.

45.2.1 Mixed States

Some formulations of quantum mechanics refer to the concept of a **mixed state**: a randomly chosen quantum state, e.g., we could say "mixed state ρ_{mom}" is "probability $\frac{1}{2}$ of $|0\rangle$, probability $\frac{1}{2}$ of $|1\rangle$" and "mixed state ρ_{dad}" is 'probability $\frac{1}{2}$ of $|+\rangle$, probability $\frac{1}{2}$ of $|-\rangle$". You first proved, given ρ_{mom}, no matter what kind of action + measurement you do, you see 0 or 1 with probabality $\frac{1}{2}$ each. And now we've proved the same, but given ρ_{dad}. **There is no physical way to distinguish at all between ρ_{mom} and ρ_{dad}.** In other words, one could argue that they are the same state (similar to how $|v\rangle$ and $-|v\rangle$ are the same). Indeed, there is a "more advanced" notation for states and mixed states in which $|v\rangle$ and $-|v\rangle$ are given identitcal mathematical expressions, but just remember that there's no algorithm for distingushing $|v\rangle$ from $-|v\rangle$ and ρ_{mom} from ρ_{dad}.

46 Discriminating 2 qubits, multiple copies

Video #046/100 Table of Contents

LawfulEvilCorp: (trustworthy, but very mean) promises $|test\rangle$ is either $|0\rangle$ or $Rot_{1.8^\circ}|0\rangle$ H : Yes or no, is $|test\rangle = Rot_{1.8^\circ}|0\rangle$? We'd measure in standard basis: if we see "1", say yes happily! if we see "0", say no sadly. This has no false positives, but lots of false negatives. The probability we correctly say "yes" is $\sin^2 1.8^\circ$.

Say this corporation is nice, and you can get numCopies of $|test\rangle$. And generalize the rotation to $Rot_{\frac{pi}{n}}$

`Distinguisher`
for $i = 1 : numCopies$,

measure ith qubit in standard basis

if outcome "1": say "yes" & halt

say no

Would numCopies being equal to n be enough?

- Case i: $|test\rangle = |0\rangle$, Distinguisher always outputs 0.
- Case ii: $|test\rangle = Rot_{\frac{\pi}{n}}|0\rangle$. Then $\mathbb{P}$(Distinguisher outputs yes) $= \sum_{i=1}^{n} \mathbb{P}$(trial i is "1")

  E(Distinguisher outputs yes in n trials) $= \sum_{i=1}^{n} \sin^2(\frac{\pi}{n}) \leq \sum_{i=1}^{n} \frac{\pi^2}{n^2} \leq \frac{10}{n}$ yeses

Now say we get numCopies= n^2: E(Distinguisher outputs yes in n^2 trials) $\leq \sum_{i=1}^{n^2} \frac{\pi^2}{n^2} = 10$ yeses.

Properly: in case ii, with n^2 trials, $\mathbb{P}$(never get "1") $= \left(1 - \sin^2\left(\frac{\pi}{n}\right)\right)^{n^2} \approx \left(1 - \left(\frac{\pi}{n}\right)^2\right)^{n^2}$ (using $\sin x = x$ for small x). Another approximation tip is that $1 + x \approx e^x$ if x is small (again, Taylor Series, $e^x = 1 + x + x^2/2! + x^3/3! + \dots$). Then we set $x = -\frac{\pi^2}{n^2}$ to get that $\mathbb{P}$(never get "1") $\approx \left(e^{-\frac{\pi^2}{n^2}}\right)^{n^2} = e^{-\pi^2} \leq 0.000052$. So $\mathbb{P}$(Distinguisher correct) $= \mathbb{P}$(get at least one "1") $= 1 - \mathbb{P}$(never get "1") ≥ 0.99948.

For that to be 100% rigorous, we ignore the approximation steps and directly show $1 - (\sin^2 \frac{\pi}{n}) \leq e^{-\frac{\pi^2}{n^2}}$. This works since $1 - (\sin^2 x) \leq e^{x^2}$ when $|x| \leq \frac{\pi}{2}$, but that's a calculus proof.
Tyler's Note: It's decently quick. We WTS $\cos^2 x \leq e^{x^2}$. By Taylor series expansion, $1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots \leq 1 + x^2 + \frac{x^4}{2!} + \frac{x^6}{3!} + \dots$ works since we can rearrange to $0 \leq x^2 + \frac{x^2}{2!} + \frac{x^4}{2!} - \frac{x^4}{4!} + \frac{x^6}{3!} + \frac{x^6}{6!} + \dots$.

So with enough copies, we can solve difficult tests. Will prove that it's impossible to get 100% certainity with just 1 qubit unless you are given perpendicular options (like in the mom and dad tests), and also work towards the no cloning principle.

46.1 Exercises: Zero-sided error

Let $0 < \theta < \frac{\tau}{4} = \frac{\pi}{2}$. Let $|u\rangle \in \mathbb{R}^2$ be the qubit state at angle $\frac{\theta}{2}$ from $|0\rangle$, and let $|v\rangle \in \mathbb{R}^2$ be the qubit state at an angle of $-\frac{\theta}{2}$ from $|0\rangle$ (thus $|u\rangle$ and $|v\rangle$ are at angle θ from each other).

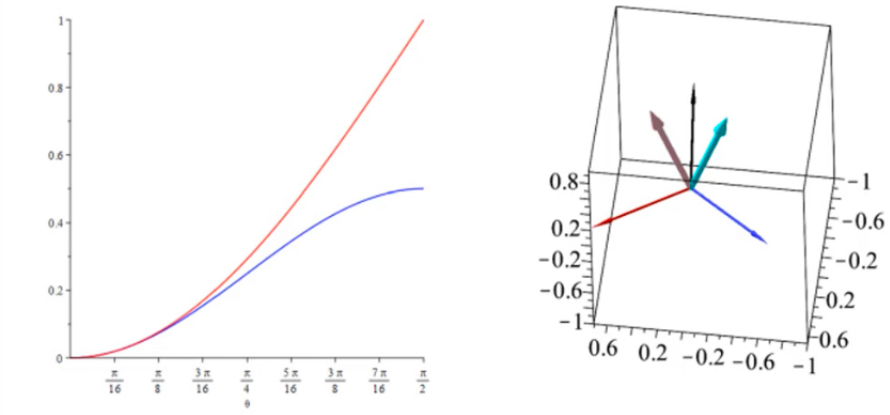
Monty hands you a qubit A, promised to be in a state $|test\rangle$ that is either $|u\rangle$ or $|v\rangle$. Your task is to guess whether $|test\rangle = |u\rangle$ or $|test\rangle = |v\rangle$. You are also allowed to *not guess*. Monty gives you \$1 if you guess correctly, but you owe monty \$1000 if you guess incorrectly. So you decide to *never* make a guess unless you're absolutely 100% sure you know the answer.

(a) Describe a strategy you can use using classical randomness (in fact, just one coin flip) and measuring in different bases, in which the probability of you making a guess is at least $\frac{1}{2}\sin^2\theta$ (and this guess is 100% sure to be correct).

Extra from Prof O'Donnell: If you think of the same solution I did, you guess with probability exactly $\frac{1}{2}\sin^2\theta$. There's something odd about this formula. If $\theta = \frac{\tau}{4} = \frac{\pi}{2} = 90°$, then there is an obvious strategy in which you guess with probability 1; yet $\frac{1}{2}\sin^2 90° = \frac{1}{2}$. In fact, this strategy is not optimal for all arbitrary θ! But to get an optimal strategy, you need to go into the third dimension.

AKA, you need to add a new qubit. This actually takes you to four dimensions, but I promise you only really need three. So feel free to ignore the $|11\rangle$ dimension. Moreover, assume that with quantum instructions it is possible to implement any rotations you want in $\mathbb{R}^3$. Thus, you can implement "Measure in the basis $\{|f\rangle, |g\rangle, |h\rangle\}$" where we have an orthonormal set of basis vectors (in particular, when you apply this measurement to some state $|v\rangle$, it displays f with probability $\langle f|v\rangle^2$, similarly for g and h).

(b) Show how you can win \$1 with probability $1 - \cos\theta$.

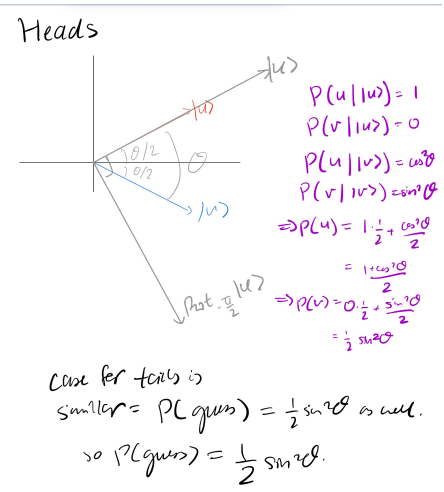


Left is plot with $\frac{1}{2}\sin^2\theta$ in blue, $1 - \cos\theta$ in red. Right is a hint for $\theta = 36°$

46.2 Solutions TODO: BONUS

(a) Flip a coin. If we see heads, measure in $\{|u\rangle, Rot_{\frac{\pi}{2}}|u\rangle\}$. If tails, measure in $\{|v\rangle, Rot_{\frac{\pi}{2}}|v\rangle\}$.

- Assume you flipped heads. (prob 1/2)
  - If you see $|u\rangle$, whatevers. $|u\rangle$ always gets measured as $|u\rangle$, but $|v\rangle$ is sometimes measured as $|u\rangle$.
  - If you see $Rot_{\frac{\pi}{2}}|u\rangle$, you know for sure $A = |v\rangle$. This occurs with $\sin^2\theta$ probability (see drawing).
- Assume you flipped tails. (prob 1/2)
  - If you see $|v\rangle$, whatevers. $|v\rangle$ always gets measured as $|v\rangle$, but $|u\rangle$ is sometimes measured as $|v\rangle$.
  - If you see $Rot_{\frac{\pi}{2}}|v\rangle$, you know for sure $A = |u\rangle$. This occurs with $\sin^2\theta$ probability.



So overall probability of guessing is $\frac{1}{2}\sin^2\theta$.

(b) if $\theta = \frac{\pi}{2}$, $|u\rangle$ and $|v\rangle$ are orthogonal, so we'd just measure in the basis $\{|u\rangle, |v\rangle\}$. Guaranteed win.

So what's a general optimal strategy? Well let's try to think of one that replicates this optimal strategy for $\theta = \frac{\pi}{2}$ that involves a second qubit.

We want to end up with probability $1 - \cos\theta$ of guessing. Since there are half angles for $|u\rangle$ and $|v\rangle$, look at the half-angle identity for cosine:

$$\cos 2x = 2\cos^2 x - 1 \implies \cos x = 2\cos^2\frac{x}{2} - 1 \implies 1 - \cos x = 2\cos^2\frac{x}{2}$$

Note that, for $|u\rangle$ and $|v\rangle$, $\langle u|0\rangle^2 = \langle v|0\rangle^2 = \cos^2\frac{\theta}{2}$

- What if $A = |u\rangle$?
- What if $A = |v\rangle$?

Going to shelve this for now.

47 Discriminating 2 qubits, optimal error (Lecture 13)

Video #047/100 Table of Contents

Recap on distinguishing 2 qubit states:

- If their angle is 90°, it's doable perfectly
- If their angle is 60°, can achieve

  25% one-sided error

  6.7% two-sided error
- For angle $\frac{\pi}{n}$, could achieve one sided success with probability $\approx \frac{10}{n^2}$

  hence high success probability given n copies of the qubit

Promise: Qubit in state $|test\rangle$ is either $ketv$ or $|w\rangle$ (two known vectors at angle θ).

Question: Yes or no, is $|test\rangle = |v\rangle$?

Requirement: 1-sided error, no false positive.

What's the lowest possible error probability (for false negatives)?

Algorithm idea: measure in basis $\{|w\rangle, |w^{\perp}\rangle\}$. Say "yes" if outcome is $w^{\perp}$, else say no.

- No false positives
- Error when $|test\rangle = |v\rangle$ is $\cos^2\theta$

We claim there is no better algorithm. This is because the requirement means that one of our measurement bases has to perfectly line up with one of the outcomes. And any unitary operation preserves the angle θ between the two possible states, so you can never get a better probability than $\cos^2\theta$ because your other basis vector for measurement must be perpendicular to the first one (which you already assigned to one of the states).

Even if you rotate/reflect $|test\rangle$, or add some new qubits then do some unitary ops in higher dimensions, or we could use classical randomness.

47.1 Exercises

Professor Mackey comes to you and says, "I have prepared a beautiful photon for you. I promise its state, $|test\rangle$, is either $|0\rangle$ or $Rot_{135^\circ}|0\rangle$. As you know, $135^\circ = \frac{3\pi}{4}$ radians. Question: Yes or no: is $|test\rangle = Rot_{135^\circ}|0\rangle$?"

For the following questions, you have five choices:

A. $(|0\rangle, Rot_{135^\circ}|0\rangle)$

B. $(|0\rangle, |1\rangle)$

C. $(|0\rangle, Rot_{45^\circ}|0\rangle)$

D. $(Rot_{22.5^\circ}|0\rangle, Rot_{112.5^\circ}|0\rangle)$

E. $(|+\rangle, |-\rangle)$

(a) Suppose you wish to answer Mackey's algorithm with 1-sided error. Specifically, you want a "yes means yes" style algorithm, meaning no false positives. Which of the following is a basis you can measure in to accomplish this?

(b) In your scheme from the previous problem, what is the error probability; that is, the probability of a false negative?

(c) Suppose instead you were willing to have 2-sided error (both false positives and false negatives). Moreover, suppose you were trying to have the maximum of the false-positive probability and the false-negative probability to be as small as possible. Of the following, which is the best basis to measure in?

(d) In your scheme from the previous problem, what is the worst case error probability (the max of the false-positive and false-negative probabilities)?

47.2 Solutions

(a) Note that A, C aren't even valid orthonormal bases. So we ignore them.

We would use B. Whenever we see $|1\rangle$, we know for sure $|test\rangle = Rot_{135^\circ}|0\rangle$. Thus we answer yes, and have no false positives. If we see $|0\rangle$, we always say no, but there is a chance of false negative.

(b) The probability of false negative is the probability of seeing $|0\rangle$ given $|test\rangle = Rot_{135^\circ}|0\rangle$. That's equal to $\langle 0| Rot_{135^\circ}|0\rangle$, which is $\cos^2 135^\circ = \frac{1}{2}$.

(c) Note that A and C aren't valid, and B guarantees one-sided error, so we're left with D and E. However, notice that E would also guarantee 1-sided error: since $|-\rangle$ is at angle -45°, which is 180° away from 135°, we will always measure $|-\rangle$ given $|test\rangle = Rot_{135^\circ}|0\rangle$, and we can only see $|+\rangle$ if $|test\rangle = |0\rangle$, so we cannot see false negatives.

We would use D.

(d) First we need to establish the measuring strategy. We want to minimize false negative, AKA we minimize the probability that $|test\rangle = Rot_{135^\circ}|0\rangle$ when we say "no". We also want to maximize false positive, AKA we maximize the probability that $|test\rangle = |0\rangle$ when we say "yes".

Let $|g\rangle, |h\rangle$ be $Rot_{22.5^\circ}|0\rangle, Rot_{112.5^\circ}|0\rangle$ respectively.

If we say "yes" if we see $|h\rangle$ and "no" otherwise, then

$$\begin{aligned}\mathbb{P}(FP) &= \langle h|0\rangle^2 = \cos^2 112.5^\circ \approx 0.1464\\ \mathbb{P}(FN) &= \langle g| Rot_{135^\circ}|0\rangle^2 = (\cos 22.5^\circ \cos 135^\circ + \sin 22.5^\circ \sin 135^\circ)^2\\ &= \cos^2(22.5^\circ - 135^\circ) = \cos^2 -112.5^\circ \approx 0.1464\end{aligned}$$

If we say "yes" if we see $|g\rangle$ and "no" otherwise, then

$$\begin{aligned}\mathbb{P}(FP) &= \langle g|0\rangle^2 = \cos^2 22.5^\circ \approx 0.8536\\ \mathbb{P}(FN) &= \langle h| Rot_{135^\circ}|0\rangle^2 = (\cos 112.5^\circ \cos 135^\circ + \sin 112.5^\circ \sin 135^\circ)^2\\ &= \cos^2(112.5^\circ - 135^\circ) = \cos^2 -22.5^\circ \approx 0.8536\end{aligned}$$

So actually, if we want to maximize false-positive and minimize false-negative, we'd say yes when we see $|g\rangle$, and say no when we see $|g\rangle$...? Whatever. The maximum error probability is 0.8536.

48 $|v\rangle$ and $-|v\rangle$ are indistinguishable

Video #048/100 Table of Contents

Notice that the graph of $\cos^2\theta$, which is our 1-sided error distribution, is even - it is the same for positive and negative θ. Intruguing is that failure probability goes to 100% at $\theta = \pm\pi = \pm 180°$.

No algorithm can distinguish $|v\rangle$, $-|v\rangle$ at all. In fact, $|v\rangle$ and $|-v\rangle$ are the "same state"! This can be succinctly expressed as "Global phase (sign) doesn't matter"

Warning: Don't try to exploit this fact too hard, or else you could say fallaciously that $|1\rangle$ and $-|1\rangle$ are the same, so $a|0\rangle + b|1\rangle = a|0\rangle - b|1\rangle$. But that's just not true (otherwise wtf would Hadamard be useful for).

More specific quantum notation (density matrices) get rid of this idiosyncracy.

48.1 Exercises: No Discrimination Principle

48.1.1 Setup

Let $|v\rangle$ and $|w\rangle$ be (real) 1-qubit states at angle θ. Let $\mathcal{D}$ be **any** algorithm that takes as input a qubit A, outputs "yes/no", and has the guarantee that it says "no" with 100% probability when A is in state $|w\rangle$. We would like to give a complete argument that when A is in state $|v\rangle$, the probability that $\mathcal{D}$ says "no" is at least $\cos^2\theta$. So what could $\mathcal{D}$ possibly do?

1. It could do "new qubit" several times. It doesn't *have* to do this at the beginning, but on the other hand, there's no harm is assuming it does. So we can ssume it starts with, say, "new qubit $B_1, \ldots, b_{k-1}$" for some positive integer k.
2. It could do a bunch of unitary operations to the qubits, but these could all be combined into a single composite $2^k \times 2^k$ unitary.
3. It could do "extract all". Of course, it had better do this at least one; but then after it does, there is no way for $\mathcal{D}$ to gain any more information. So we may as well assume it does "extract all" at the end.
4. Based on the string $s \in \{0,1\}^k$ displayed by "extract all", $\mathcal{D}$ must output "yes" or "no". So we can assume that there is some subset $N \subseteq \{0,1\}^k$ of displays on which it outputs "no", and on the remaining displays $Y = \{0,1\}^k \setminus N$ it outputs "yes".

So we can assume $\mathcal{D}$ looks like the following, for some $2^k \times 2^k$ unitary U and some $N \subseteq \{0,1\}^k$:

1. new qubit $B_1, \ldots, b_{k-1}$
2. apply U to $A, B_1, \ldots, b_{k-1}$
3. extract all
4. if the result s of "extract all" is in N, output "no", else output "yes"

Actually, you might consider the possibility that $\mathcal{D}$ uses classical randomness; we'll discuss this later.

48.1.2 Questions

Let $|v'\rangle \in \mathbb{R}^{2^k}$ denote the state of the qubits after line 2 assuming that A starts in state $|v\rangle$, and similarly define $|w'\rangle$ assuming A starts in $|w\rangle$.

(a) Argue that $\langle v'|w'\rangle = \cos\theta$. (Note: in case I never made it fully clear, if a qubit is in state $x|0\rangle + y|1\rangle$, and then you do e.g. "new qubit B_1, B_2", the new state is $x|000\rangle + y|100\rangle$.)

(b) Argue that if $|w'\rangle$ has nonzero amplitude on some basic state $|s\rangle$ for $s \in \{0,1\}^k$, then $s \in N$.

(c) Argue that indeed $\mathcal{D}$ outputs "no" with probability at least $\cos^2\theta$ when A is initially $|v\rangle$. (Be careful, it is not the case that this probability is necessarily *exactly* $\cos^2\theta$.)

(d) Returning to the possibility that $\mathcal{D}$ uses classical randomness, this would mean that $\mathcal{D}$ is something like, "with probability .3, do $\mathcal{D}_1$, with probability .6 do $\mathcal{D}_2$, with probability .1 do $\mathcal{D}_3$", where $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ are "deterministic" algorithms of the type you reasoned about in parts (a)-(c). Explain why an algorithm $\mathcal{D}$ like this *still* must output "no" with probability at least $\cos^2\theta$ when A is initially $|v\rangle$.

48.2 Solutions

(a) Since $|v\rangle$ and $|w\rangle$ are at angle θ, then $\langle v|w\rangle = \cos\theta$. Unitary operations perserve angle. New qubit doesn't affect angle either. So $|v'\rangle$ and $|w'\rangle$ are also at angle θ. Then $\langle v'|w'\rangle = \cos\theta$.

More formally for new qubit, let $|v\rangle = x|0\rangle + y|1\rangle$ and $|w\rangle = a|0\rangle + b|1\rangle$. After new qubits, we get states $x|00\ldots0\rangle + y|10\ldots0\rangle, a|00\ldots0\rangle + b|10\ldots0\rangle$ - the inner product $x^\dagger a + y^\dagger b$ is preserved, so the angle θ is too.

(b) We said that once we extract, we output "no" with 100% probability when A is in state $|w\rangle$. Then once we extract after line 2, we extract from the state $|w'\rangle$. The initial stipulation means that any possible state s we see from this extract is in N. But we only see s if and only if there is nonzero amplitude on s. Thus, s has nonzero amplitude in $|w'\rangle \implies s \in N$.

(c) The probability we output "no" when A is initially $|v\rangle$ is at least the probability of measuring a state with nonzero amplitude in $|w'\rangle$ given the state is $|v'\rangle$. Thus, the probability of this false negative result is $\langle v'|w'\rangle^2 = \cos^2\theta$ at least.

We say "at least" beacuse part (b) is not a biimplication - there may be states $s \in \{0,1\}^k$ that are in N that have nonzero amplitude in $|w'\rangle$. AKA, it is not guaranteed that $s \in N \implies s$ has nonzero amplitude in $|w'\rangle$.

(d) We proved that in part (c), the state $|v'\rangle$ (even as the result of classical randomness), is going to lead to an output of "no" with probability at least $\cos^2\theta$. There is no way to decrease this probability, since the number of states in N is strictly (lower) bounded by the states in $|w'\rangle$ with nonzero amplitude - otherwise, we couldn't guarantee with 100% confidence that $\mathcal{D}$ outputs "no" when A starts in $|w\rangle$.

More formally, each deterministic $\mathcal{D}_i$ needs to maintain the "no false positives" constraint. Then each $\mathcal{D}_i$ needs to include all states $s \in \{0,1\}^k$ where s has nonzero amplitude in $|w'\rangle$. So

$$s \text{ has nonzero amplitude in } |w'\rangle \implies s \in N_i$$

where N_i is the states that are possible with output "no" after $\mathcal{D}_i$. By similar logic from part (c), each $\mathcal{D}_i$ has probability of at least $\cos^2\theta$ of outputting no when A starts as $|v\rangle$, so we still have that lower bound of probability $\cos^2\theta$ of outputting "no" when A is initially $|v\rangle$.

49 The No-Distinguishing Principle =¿ No-Cloning Principle

Video #049/100 Table of Contents

No-Distingushing Principle: (for non-perpendicular states): As long as $\theta \neq \pm 90°$, there is no perfect distinguisher. E.g., any algorithm for $|0\rangle$ vs $|+\rangle$ with 1-sided error has error $\geq \frac{1}{2} = \cos^2 45°$ (since $45°$ is the angle between $|0\rangle$ and $|+\rangle$).

If you can get m copies of $|test\rangle$, you can reduce the false negative probability to $(\frac{1}{2})^m$.

What follows is the **No-Cloning Principle:** There's no physical device (quantum algorithm) that takes as input one copy of a mystery state $|test\rangle$ and then oumps out a stream of particles all in state $|test\rangle$. Can't even make one more copy.

Prove by contrapositive that No-Distingushing Principles implies No-Cloning Principle: Suppose we could clone. Given 1 copy of $|test\rangle$, promised to be $|0\rangle$ or $|+\rangle$, we could distinguish with error $<< \frac{1}{2}$. Just clone $|test\rangle$ many (m) times and do the distinguisher algorithm (which as stated above, once we have m copies, we can reduce false negative probability below $\frac{1}{2}$).

Can't even succeed in cloning when the promised input is $|0\rangle$ or $|+\rangle$. Does work when the promised input is $|0\rangle$ and $|1\rangle$; in that case, you have a perfect distinguisher (since they are perpendicular) and you'd just make a bunch of new qubits and then decide whether to toggle or not.

You could also say that No-Cloning implies No-Learning - given one copy of a state $|test\rangle$, learning the amplitudes is impossible. They're kind of an if and only if. If you had a bunch (like a trillion) copies of an unknown state, you could just keep measuring in different bases and eventually figure out the amplitudes to arbitrary accuracy. If you could learn a state's angle perfectly, then obviously you could clone it by just making a new qubit and doing the right rotation.

No-Cloning isn't inherently a quantum thing - it's like a sticky coin. Once you've flipped it once and "measured" it, it's stuck and can't be remeasured again. So you can't learn the probability it lands on heads or tails. Only sticky coin you can distinguish is a 100% heads 0% tails coin (or 0% heads and 100% tails; just see what side it lands on). So while probability and quantum states aren't exactly the same, the underlying intuition is the same.

49.1 Exercises: Statistics I

This problem is not related to quantum per se; instead it relies on your prerequisite knowledge of basic "classical" probability/statistics. But it will be important to have these facts at your fingertips, so ew can compare with the "quantum" probability/statistics that will arise in our deeper investigations.

Suppose you have a "p-biased" coin, which, when you flip it, comes up "Heads" with probabiliy p and "Tails" with probability $1 - p$. Intuitively, think of p is very "small".

(a) Suppose you flip the coin $\lfloor \frac{0.01}{p} \rfloor$ times. Prove that the probability of getting at least 1 Heads is at most 1%.

(b) Suppose on the other hand you flip the coin $\lceil \frac{5}{p} \rceil$ times. Prove that the probability of *not* getting at least 1 Heads is at most 1%. You can and should use the fact that $1 - x \leq e^{-x}$ for all x.

Tyler's Note: observe the ceiling here, not floor.

49.2 Solutions

Proof. Helpful facts:

(I) $0 < p \leq 1$. If probability p were zero, the number of flips in parts (a) and (b) would be undefined.

a. It follows that, for any positive x, $px \leq x$ and $(1-p)x < x$.

b. From 1a, we derive $x \leq \frac{x}{p}$ and $x < \frac{x}{1-p}$

(II) $1 - x \leq e^{-x}$ for all x

(III) Definitions of floor and ceil:

a. $\lfloor \frac{x}{p} \rfloor \leq \frac{x}{p}$ for all x (defn of floor)

b. $\lceil \frac{x}{p} \rceil \geq \frac{x}{p}$ for all x (defn of ceil)

c. $\lceil \frac{x}{p} \rceil - \lfloor \frac{x}{p} \rfloor = 1$ for $x \neq 0$ (defn of floor and ceil)

(IV) Follows from (3):

a. $(1-p)^{\lfloor \frac{x}{p} \rfloor} \leq e^{-x}$

b. $(1-p)^{\lceil \frac{x}{p} \rceil} \geq e^{-x}$

$$
\begin{aligned}
\textit{Proof:}\quad & (1-p)^{\lfloor \frac{x}{p} \rfloor} \leq (1-p)^{\frac{x}{p}} \text{ and } (1-p)^{\lceil \frac{x}{p} \rceil} \geq (1-p)^{\frac{x}{p}} && \text{(III.a, III.b)} \\
\implies & (1-p)^{\lfloor \frac{x}{p} \rfloor} \leq \left(e^{-p}\right)^{\frac{x}{p}} \text{ and } (1-p)^{\lceil \frac{x}{p} \rceil} \geq \left(e^{-p}\right)^{\frac{x}{p}} && \text{(II)} \\
\implies & (1-p)^{\lfloor \frac{x}{p} \rfloor} \leq e^{-p\left(\frac{x}{p}\right)} \text{ and } (1-p)^{\lceil \frac{x}{p} \rceil} \geq e^{-p\left(\frac{x}{p}\right)} && \text{(arithmetic)} \\
\implies & (1-p)^{\lfloor \frac{x}{p} \rfloor} \leq e^{-x} \text{ and } (1-p)^{\lceil \frac{x}{p} \rceil} \geq e^{-x} && \text{(Q.E.D.)}
\end{aligned}
$$

(V) Given n flips, $\mathbb{P}(\text{Heads} = 0) = (1-p)^n$ and $\mathbb{P}(\text{Heads} \geq 1) = 1 - \mathbb{P}(\text{Heads} = 0)$.

Proof of (a): WTS $1 - (1-p)^{\lfloor \frac{0.01}{p} \rfloor} \leq 0.01$ (rephrase problem using V)

$$
\begin{aligned}
& (1-p)^{\lceil \frac{0.01}{p} \rceil} \geq e^{-0.01} && \text{(IV.b)} \\
\implies & \frac{(1-p)^{\lceil \frac{0.01}{p} \rceil}}{1-p} = (1-p)^{\lceil \frac{0.01}{p} \rceil - 1} \geq \frac{e^{-0.01}}{1-p} && \text{(arithmetic)} \\
\implies & (1-p)^{\lfloor \frac{0.01}{p} \rfloor} \geq \frac{e^{-0.01}}{1-p} && \text{(III.c, clearly } \tfrac{0.01}{p} \neq 0\text{)} \\
\implies & (1-p)^{\lfloor \frac{0.01}{p} \rfloor} > e^{-0.01} && \text{(I.b)} \\
\implies & 1 - (1-p)^{\lfloor \frac{0.01}{p} \rfloor} < 1 - e^{-0.01} && \text{(arithmetic)} \\
\implies & 1 - (1-p)^{\lfloor \frac{0.01}{p} \rfloor} < 0.009950 && \text{(} 1 - e^{-0.01} \approx 0.009950\text{)} \\
\implies & 1 - (1-p)^{\lfloor \frac{0.01}{p} \rfloor} < 0.01 && \text{(Q.E.D.)}
\end{aligned}
$$

Proof of (b): WTS $(1-p)^{\lceil \frac{5}{p} \rceil} \leq 0.01$ (rephrase problem using V)

$$
\begin{aligned}
& (1-p)^{\lfloor \frac{5}{p} \rfloor} \leq e^{-5} \approx 0.0067 && \text{(IV.a)} \\
\implies & (1-p)^{\lfloor \frac{5}{p} \rfloor} \times (1-p) = (1-p)^{\lfloor \frac{5}{p} \rfloor + 1} \leq e^{-5} \times (1-p) && \text{(arithmetic)} \\
\implies & (1-p)^{\lceil \frac{5}{p} \rceil} \leq e^{-5} \times (1-p) && \text{(III.c, clearly } \tfrac{5}{p} \neq 0\text{)} \\
\implies & (1-p)^{\lceil \frac{5}{p} \rceil} < e^{-5} && \text{(I.a)} \\
\implies & (1-p)^{\lceil \frac{5}{p} \rceil} < 0.006738 && \text{(} e^{-5} \approx 0.006738\text{)} \\
\implies & (1-p)^{\lceil \frac{5}{p} \rceil} < 0.01 && \text{(Q.E.D.)}
\end{aligned}
$$

We have proven parts (a) and (b). □

50 Discriminating two rotations

Video #050/100 Table of Contents

Recall: Given $|test\rangle$ either $|0\rangle$ or $Rot_{\frac{\pi}{n}}|0\rangle$

With one copy, 1-sided success probability $\approx \frac{10}{n^2}$.
n copies brings this to $\approx \frac{10}{n}$.
n^2 copies brings this to ≤ 0.00006

What if you break into LawfulEvilCorp's office and steal the test generation machine?

$$|0\rangle \to \text{test generation machine} \to |test\rangle$$

AKA, you are able to get their source code for

`MysteryRotation` A

1. Rotate $\frac{\pi}{n}$ on A or do nothing.

Find out which one it is!

New move you can do now that you've stolen the test generation box: put in qubits that aren't $|0\rangle$.

`RotationDetective`

1. new qubit A
2. for $t = 1 \ldots \frac{n}{2}$, do MysteryRotation(A)

   note this is in $O(n)$, as opposed to the $O(n^2)$ copies when we didn't have the test generation box
3. extract all, print "yes" if "1", else print "no"

Claim: 100% Success! Observe $\frac{\pi}{n} \cdot \frac{n}{2} = \frac{\pi}{2} = 90°$. So basically, we send in a qubit, then send it back in, $\frac{n}{2}$ times. Then we extract it.

- Case "yes": Then our qubit's angle with $|0\rangle$ went from $0 \to \frac{\pi}{n} \to \frac{2\pi}{n} \to \cdots \to \frac{n-1}{2} \cdot \frac{\pi}{n} \to \frac{n}{2} \cdot \frac{\pi}{n} = \frac{\pi}{2}$. So when we extract in the standard basis, we'll see "1"!
- Case "no": Then our qubit's angle with $|0\rangle$ just stayed at 0 the entire time, so when we extract in the standard basis, we'll see "0"!

So we've got a quadratic speedup to let us do "Rotation Estimation" (really called Phase Estimation Algorithm, "phase" because complex numbers could be involved, and it's the most important subroutine in quantum algorithms).

But this heavily relies on the promise that test generation box is strictly defined by MysteryRotation. Designing quantum algorithms is trying to get yourself into this scenario, when you know a piece of code is either doing A or B, and you want to figure out which one it's doing.

Additional technical point: We know that any one qubit unitary is either a rotation or reflection.

`MysteryReflection` A

1. Reflect across $|0\rangle$ (AKA if A then minus) or reflect across $\theta = \frac{\pi}{2n}$.

We don't do the same thing we did to solve MysteryRotation; instead, throw in a known reflection first - and recall that two reflections make a rotation.

`ReflectionDetective`

1. new qubit
2. reflect through $|0\rangle$ (if A then minus)
3. MysteryReflection

Notice here: if MysteryReflection just does a reflection across $|0\rangle$, we'll always see "0" in the output and say "no". If MysteryReflection is reflecting across $\theta = \frac{\pi}{2n}$, then ReflectionDetective becomes a composite instruction that is a rotation by $2\theta = \frac{\pi}{n}$. Now, we'll just do Rotation detective, except we replace MysteryRotation with ReflectionDetective!

50.1 Exercises

You're still recording videos of yourself explaining all your solutions to the Exercises, right?

50.2 Solutions

No... I'll record them all at the end I guess.

51 The Elitzur-Vaidman Glitter Bomb

Video #051/100 Table of Contents

Bomb problem: imagine you make traps that consist of an opaque sealed box that has a tiny slit that is only wide enough to fit one photon. There's a horizontal polarizing filter inside that's attached to a highly sensitive wick. If the measurement outcome on the HPF is 0, the photon goes thorugh. If the measurement outcome is 1, the photon will get blocked, become heat, and set off the bomb.

We want to know if one of the sealed boxes is a loaded bomb or empty (presumably, without setting off the bomb).

You could,

- put a photon in $|1\rangle$. But this is too risky! You'll definitely explode in case bomb.
- put a photon in $|0\rangle$. Well in either case, the photon passes through.
- put a photon in $|+\rangle$.
  - Case Empty: $|+\rangle$ comes out.
  - Case Bomb: akin to measuring $|+\rangle$ in standard basis. so there's a 50% chance of it exploding. But if it doesn't explode, $|0\rangle$ comes out.
    could do $|0\rangle$ vs $|+\rangle$ 1-sided error test (bomb means bomb), which has $\mathbb{P}(\text{false negative}) = \frac{1}{2}$. But a better algorithm: let $n = 10^6$ (depending on how much time you have).
    `MysteryBoxRotation(A)`
    1. Rotate A by $\frac{\pi}{n}$
    2. Box(A); either explodes or spits out a (maybe new) version of A.
  - Case Empty: MysteryBoxRotation is just rotate by $\frac{\pi}{n}$.
  - Case Bomb: if you put $A = |0\rangle$, acts like do nothing on $|0\rangle$, but
    there might be an explosion $\mathbb{P} = \sin^2\left(\frac{\pi}{n}\right)^2$ after you rotate A by $\frac{\pi}{n}$.
    If no boom, it'll come out as $|0\rangle$. Then we know there's a bomb.

So here's what you do:
`BombDetective`

1. new qubit A
2. for $t = 1 \ldots \frac{n}{2}$, do MysteryBoxRotation(A)

   note this is in $O(n)$, as opposed to the $O(n^2)$ copies when we didn't have the test generation box
3. extract all, print "empty" if "1", else print "bomb"

Suppose the box is empty. Then BombDetective just ends up rotating A by $\frac{\pi}{2}$, so "1" comes out. But suppose the box has a bomb. Then if you survive, BombDetective basically just does nothing to $|0\rangle$ over and over again, so "0" comes out.

The downside is that there's a small chance of explosion. But what's the probabilty of explosion? The chance we explode at all is the chance we explode at $t = 1, 2, \ldots \frac{n}{2}$. At each time, there's a $\sin^2 \frac{\pi}{n}$ chance of exploding, but since $\frac{\pi}{n}$ is small, $\sin^2 \frac{\pi}{n} \approx \frac{\pi^2}{n^2} \approx \frac{10}{n^2}$. Over $\frac{n}{2}$ iterations, that's $\frac{10}{n^2} \cdot \frac{n}{2} = \frac{5}{n}$ probability of exploding. Which for really big n (like our example, $n = 10^6$, that's quite a small chance you explode).

It's a time versus safety tradeoff. The bigger n you choose, the more time it takes to run BombDetective. But bigger n means even lower chance of exploding, as $\mathbb{P}(\text{explode}) \propto \frac{1}{n}$

51.1 Exercises

Your teacher hands you a small black box, which takes in one qubit, performs some unitary on it, and then outputs the qubit. Your teacher promises you that the 1-qubit unitary implemented by the box is a *reflection: either* through the line that makes angle 45.5° with the $|0\rangle$-axis, or through the line that makes angle 44.5° with the $|0\rangle$-axis.

You can use the black box up to 45 times, after which it will stop working. You are supposed to say "yes" if it's reflecting through the 45.5° line, or say "no" if it's reflecting through the 44.5° line.

Your goal is to minimize the probability of guessing wrongly; you're trying to get both "false positives" and "false negatives" to have probability at most p, for the smallest possible p.

51.2 Solutions

`ReflectionDetective(A)`

1. reflect A through 44.5°
2. BlackBox(A)

Then in essence, ReflectionDetective is either a $2 \cdot (45.5° - 44.5°) = 2°$ rotation or a do-nothing.

`RotationDetective`

1. new qubit A
2. Rotate A by 44.5°
3. for $t = 1 \dots 45$, do RotationDetective(A)
4. Rotate A by $-44.5°$
5. extract all, print "yes" if "1", else print "no"

Thus, we use all 45 calls to BlackBox through 45 calls to ReflectionDetective.

- Case 44.5°: Then $\theta(A)$ goes from $0° \to 44.5° \to \cdots \to 44.5° \to 0°$. So A ends in $|0\rangle$. So if we extract and see "0", we say no.
- Case 45.5°. Then $\theta(A)$ goes from $0° \to 44.5° \to 46.5° \to 48.5° \to \dots 134.5° \to 90°$. So A ends in $|1\rangle$. So if we extract and see "1", we say yes.

We minimize both false positives and false negatives. In fact, we know our answer is correct.

52 Reflecting through a vector (Lecture 14)

Video #052/100 Table of Contents

Every unitary breaks down into 2-D planes where it goes rotations and 1-D lines where it goes nothing (Video #038/100). Reflection across a vector $|v\rangle$ has no 2-D planes of rotation, one axis of no-op $|v\rangle$, and all the other directions get negated.

Example in 2^n dimensions: if $(B_1 \vee B_2 \vee \cdots \vee B_n)$ then minus. Easy to program; can be coded in 2^n operations in AND/OR/NOT code (Video #026/100). Maps $|00\ldots0\rangle$ to itself. Maps any other bitstring to minus itself (negates the amplitude). This is therefore equivalent to reflection across $|00\ldots0\rangle$, or if $\perp$ to $|00\ldots0\rangle$ then minus.

Question: is it easy to code if $\perp$ to $|v\rangle$ then minus for other $|v\rangle$'s?
Answer: Yes, if you can easily create $|v\rangle$.

1. new qubit $B_1 \ldots B_n$
2. some code here (corresponding to some unitary U that is $2^n \times 2^n$) now the qubits are in state $|v\rangle$.

Then $|v\rangle = U|00\ldots0\rangle$. Hence, we can run the undo of U to $|v\rangle$ to get back to $|00\ldots0\rangle$.
AKA $|00\ldots0\rangle = U^\dagger|v\rangle$.

In order to create if $\perp$ to $|v\rangle$ then minus,

1. undo U (rotates space, brings $|v\rangle$ to $|00\ldots0\rangle$)
2. if $\perp$ to $|00\ldots0\rangle$ then minus
3. do U (rotates space back)

If $|v\rangle$ is a basic state, there's an easier way to do this. Say $|v\rangle = |101\rangle$. Then we'd just do if not B_1 or B_2 or not B_3 then minus (basically you hardcode it).

52.1 Exercises: Project and Reflect

Let $|\psi\rangle$ and $|\phi\rangle$ be two unit vectors in $\mathbb{R}^d$. For intuition's sake, you can think of d as 2 or 3. Also, it would be fine if these were complex unit vectors, but let's keep it real. We will be interested in $Q = |\phi\rangle\langle\psi|$, which is a $d \times d$ matrix, and can therefore be thought of as a linear transformation on d-dimensional vectors.

(a) Explicitly work out the matrix Q in the case $|\psi\rangle = |1\rangle$ and $|\phi\rangle = |+\rangle$, and also in the opposite case $|\psi\rangle = |+\rangle$ and $|\phi\rangle = |1\rangle$

(b) Hand-draw the expression $|\phi\rangle\langle\psi|$. Here's the trick:

- start by drawing the first bar and ϕ, like so: $|\phi$
- next, draw an X, like so: $|\phi$X
- finally, draw the ψ and final bar, forming $|\phi$ X $\psi|$ which kind looks like $|\phi\rangle\langle\psi|$.

If you try to draw the $\langle$ and $\rangle$ separately, you'll never get the points to match up. Congratulations, you now know a key secret of the quantum club :)

(c) Fill in the blanks: The transformation Q maps the vector $|\psi\rangle$ to the vector _____ and maps every vector which is orthogonal (perpendicular) to $|\psi\rangle$ to the vector _____

(d) Suppose now that $|\psi\rangle = |\phi\rangle$. Let $P = |\psi\rangle\langle\phi|$. Describe in (geometric) words what the linear transformation P does.

(e) Let $\mathbb{1}$ denote the identity matrix in $\mathbb{R}^d$ (traditionally, we'd use I, the matrix with 1's on the diagonal and 0's elsewhere). Describe in (geometric) words the transformation $\mathbb{1} - 2P$. Your description should include the words "plane perpendicular to" (plane means line in 2-dimensions and "hyperplane" in 4+ dimensions). Draw a 2-dimensional picture to assist with your description.

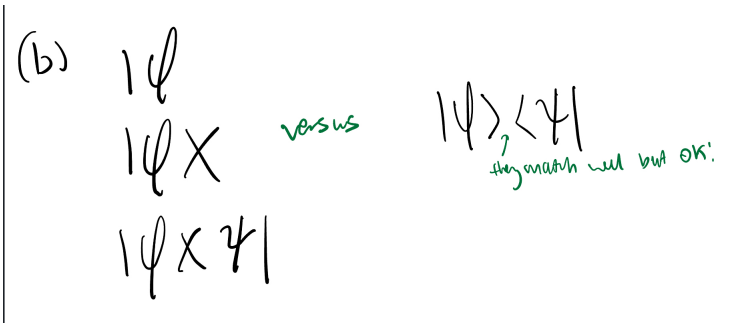
52.2 Solutions

(a) This is only reasonable if we let $d = 2$.

$|\psi\rangle = |1\rangle = (0,1)$ and $|\phi\rangle = |+\rangle = \left(\sqrt{\frac{1}{2}}, \sqrt{\frac{1}{2}}\right)$. Then $Q = |\phi\rangle\langle\psi| = \begin{bmatrix}\sqrt{\frac{1}{2}} \\ \sqrt{\frac{1}{2}}\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix} = \begin{bmatrix}0 & \sqrt{\frac{1}{2}} \\ 0 & \sqrt{\frac{1}{2}}\end{bmatrix}$

$|\psi\rangle = |+\rangle = \left(\sqrt{\frac{1}{2}}, \sqrt{\frac{1}{2}}\right)$ and $|\phi\rangle = |1\rangle = (0,1)$. Then $Q = |\phi\rangle\langle\psi| = \begin{bmatrix}0 \\ 1\end{bmatrix}\begin{bmatrix}\sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}}\end{bmatrix} = \begin{bmatrix}0 & 0 \\ \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}}\end{bmatrix}$

(b)

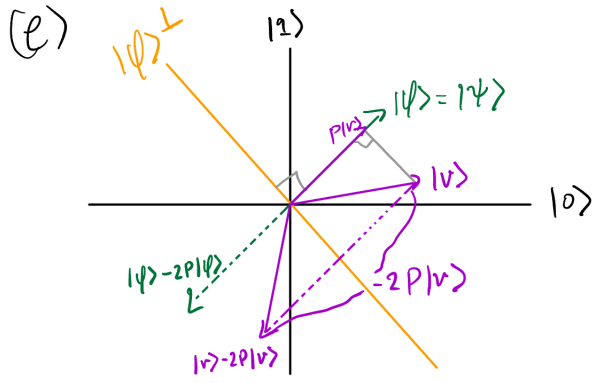


(c) $|\phi\rangle$, $|0\ldots0\rangle$.

Why? $Q = |\phi\rangle\langle\psi|$. So $Q|v\rangle = |\phi\rangle\langle\psi|v\rangle$. When $|v\rangle = |\psi\rangle$, we get $Q|v\rangle = |\phi\rangle\langle\psi|\psi\rangle = |\phi\rangle\||\psi\rangle\|^2 = |\phi\rangle(1)^2 = |\phi\rangle$. When $|v\rangle \perp |\psi\rangle$, we get $Q|v\rangle = |\phi\rangle\langle\psi|v\rangle = |\phi\rangle(0) = |0\ldots0\rangle$.

(d) P projects vectors onto $|\phi\rangle$.

(e) $\mathbb{1} - 2P$ reflects $|v\rangle$ across the plane perpendicular to $|\phi\rangle$.



53 SAT

Video #053/100 Table of Contents

Grover's Algorithm:

Recall Bias-Busting Problem Video #029/100. SAT is similar: Given classical AND/OR/NOT code C, computing $f : \{0,1\}^n \to \{0,1\}$, does there exist an input $x^* \in \{0,1\}^n$ such that $f(x^*) = 1$?

- Such an x^* satisfies f, hence "SATisfiability" problem.
- This is technically Circuit-SAT, but it's essentially the same as the SAT problem.
- It is the canonical NP-Complete problem (15-251).

The baseline algorithm for SAT is to just run f on all possible bitstrings of length n. If C contains m lines of code, then it has $O(2^n \cdot m)$ runtime, which is about 2^n (usually, the bottleneck is the number of bitstrings).

Fact: No one knows a faster algorithm, i.e. No one knows $O(1.9999^n)$ time algorithm. "$P \neq NP \iff$ no poly(n) time algorithm for SAT".

Strong Exponential Time Hypothesis ("SETH"): there is no $O(1.99\ldots9^n)$ time algorithm for any # of 9's.

There are some other variants of SAT, like Unique-SAT-Search: you're promised there's exactly one satisfying string x^*. The task is to find x^*. Grover's Algorithm solves this variant of SAT in $O(\sqrt{2}^n)$ time.

Some examples of problems that are hard to solve:

- Let $C_1(x_1, x_2, \ldots, x_{1024})$. Let $y =$ num whose binary representation is $x_1 x_2 \ldots x_{512}$ and $z =$ num whos binary representation is $x_{513} x_{514} \ldots x_{1024}$. Let $w = y \cdot z$. If $y \neq 1$ and $z \neq 1$ and $w =$ 13506[...300 digits...]7563 : return 1. Else return 0.

  w is RSA-1024 (famous factoring challenge number). Note that this exactly isn't really unique SAT. They came up with w by choosing huge numbers and multiplying them together. Hard to solve, easy to generate new hard problems.

- $C_2(x_1 \ldots, x_{32})$ $hash :=$ SHA256(SHA256(92084...0122 + x)) where the 92...number is the "block header". If hash $< 2^{192}$, return 1. Else return 0.

  This is basically what bitcoin mining does.

- $C_3 x_1, \ldots, x_{100{,}000{,}000}$. Return LeanProofChecker("$P \neq NP$ because x")

  Finding a SAT solver that would find a 10 MB or less proof that $P \neq NP$.

- $C_4(x_1, \ldots, x_{1{,}000{,}000{,}000})$ where each x_i is a parameter for a neural net. Compute the neural net's training error, return 1 is training error $< 0.1\%$, else return 0.

53.1 Exercises

Let's now show that circuit-SAT $\leq^P$ Unique-Circuit-SAT (using randomness). Given an AND/OR/NOT circuit C with n input bits, let $S(C)$ denote the number of strings $x \in \{0,1\}^n$ such that $C(x) = 1$. Recall that "solving Unique-Circuit-SAT" means finding a (in fact *the*) satisfying string for C under the promise that $S(C) = 1$.

Suppose for a given C we somehow have the inkling that $2^{42} \leq S(C) < 2^{43}$. Now say we pick a random bistring $z \in \{0,1\}^n$, and form a circuit C'_z that implements the following pseudocode:

On input $x \in \{0,1\}^n$:

Let $w = \text{SHA256}(x \oplus z)$.

Return 1 if $C(x) = 1$ and the first 42 bits of w are all 0's; otherwise, return 0.

Here, think of SHA256 as some ealy-to-implement, deterministic function that nevertheless pretty much produces "random" output strings on each string.

- Intuitively, (you don't have to be completly rigorous), explain how and why you could use the ability to solve Unique-Circuit-SAT efficiently to also solve Circuit-SAT efficiently on C, *assuming* your inkling was correct. (You might have to use your Unique-Circuit-SAT ability a bunch of times).
- Where would you get that inkling? Well, how many different inklings can there be? Intuitively, explain how to solve Circuit-SAT in poly(n) time in general, given the ability to solve Unique-Circuit-SAT in poly(n) time. (You may assume the existence of a SHA256-like function wth any number of desired output bits).

53.2 Solutions

The probability the first 42 bits are all zero for a random bitstring w in C'_z is $\left(\frac{1}{2}\right)^{42} = 2^{-42}$. The probability $C(x) = 1$ for some bitstring x is $\frac{S(C)}{n} \approx \frac{2^{42}}{n}$, so the probability C'_z outputs "1" is about $\frac{1}{n}$, which makes C'_z a Unique-Circuit-SAT problem.

- If we can solve Unique-Circuit-SAT efficiently, then we can solve C'_z efficiently. We'd have to do this a bunch of times, but eventually, we'll find an input x for which $C'_z(x)$ outputs "1". Then we'd have an x such that $C(x) = 1$, so we're done.
- You'd do the process outlined in the problem setup, for a bunch of different potential inklings.

  $(C_k)'_z$ is the program that tests the inkling $2^k \leq S(C) < 2^{k+1}$, for $0 \leq k \leq n-1$. Thus there are n values of k, so there are n inklings to test.

  On input $x \in \{0,1\}^n$:

  Let $w = \text{SHA256}(x \oplus z)$.

  Return 1 if $C(x) = 1$ and the first k bits of w are all 0's; otherwise, return 0.

  For the right value of k, we'd get the setup from the previous part of the problem, where the probability $(C_k)'_z$ outputs "1" is the product that $C(x) = 1$ and the first k bits of w are all 0's, which is equal to $\frac{S(C)}{n} \cdot \frac{1}{2^k} = \frac{1}{n}$, making $(C_k)'_z$ a Unique-Circuit-SAT problem.

  Because we can solve Unique-Circuit-SAT in poly(n) time, and we are testing n intervals, then we solve Circuit-SAT in n times poly(n) time, which is still poly(n) time.

54 Grover's Algorithm, Part 1

Video #054/100 Table of Contents

Grover 1996: A quantum algorithm solving (unique)-SAT-(search) problem in $\sqrt{2}^n$ time. Importantly, this means quantum computers break SETH (Video #053/100).

0. Make "if $f(B_1, \ldots, B_n)$ then minus" code.

   D-dimensional unitary, $D = 2^n$. One dimension of negation $|x^*\rangle$, remaining dimensions are no-ops.

   We're only negating one entry from our input,

   so it's reflection through the $D-1$ dimensional plane perpendicular to $|x^*\rangle$.

   Let's just start with Bias-Busting code Video #029/100

1. new qubit $B_1 \ldots B_n$
2. Hadamard all
3. if $f(B_1, \ldots, B_n)$ then minus
4. ...

After line 1, we have state $|00\ldots0\rangle = (1, 0, \ldots, 0)$.
After line 2, we have a uniform superposition on all n entries, $|unif\rangle = \sqrt{\frac{1}{D}}(1, 1, \ldots, 1)$.
After line 3, we are in a state where $|f\rangle = \sqrt{\frac{1}{D}}(1, 1, \ldots, 1, -1, 1, \ldots, 1)$ where the -1 is in the entry corresponding to $|x^*\rangle$. This is basically the ±1 truth table of f, hence why we called it $|f\rangle$.

Note: the states after line 2 and line 3 only differ by one coordinate. $|unif\rangle - |f\rangle = \sqrt{\frac{1}{D}}(0, 0, \ldots, 0, 2, 0 \ldots, 0)$ where the 2 is in the entry corresponding to $|x^*\rangle$.

Takeaways:

- This difference vector is very short; difference vector has length $\frac{2}{\sqrt{D}}$
- $|unif\rangle$ and $|f\rangle$ are very close, but not parallel. So there is some 2-D plane in $\mathbb{R}^D$ that contains $|unif\rangle$ and $|f\rangle$.
- By linearity, $|x^*\rangle$ is also in this 2-D plane in $\mathbb{R}^D$.
- Note that in such a plane, $|unif\rangle$ and $|f\rangle$ have angle θ between them. But the difference vector is so small, and $|unif\rangle$ and $|f\rangle$ are so close, that the difference vector is basically the arc length between $|unif\rangle$ and $|f\rangle$. Hence $\theta \approx \frac{2}{\sqrt{D}}$.
- Note $|x^*\rangle$ is in the same direction as the difference vector, just parallel in the same plane and starting at the origin, and also scaled up to make $|x^*\rangle$ a unit vector.
- To be continued!

54.1 Exercises

Consider the following operation on qubits $B_1 \ldots B_n$

$$\text{if } (B_1 \wedge B_2 \wedge \cdots \wedge B_n) \text{ then minus}$$

Geometrically, how would you describe this?

A. Reflection across the vector $|00\ldots0\rangle$

B. reflection across the $2^n - 1$ dimensional plane perpendicular to $|00\ldots0\rangle$

C. Reflection across the vector $|11\ldots1\rangle$

D. reflection across the $2^n - 1$ dimensional plane perpendicular to $|11\ldots1\rangle$

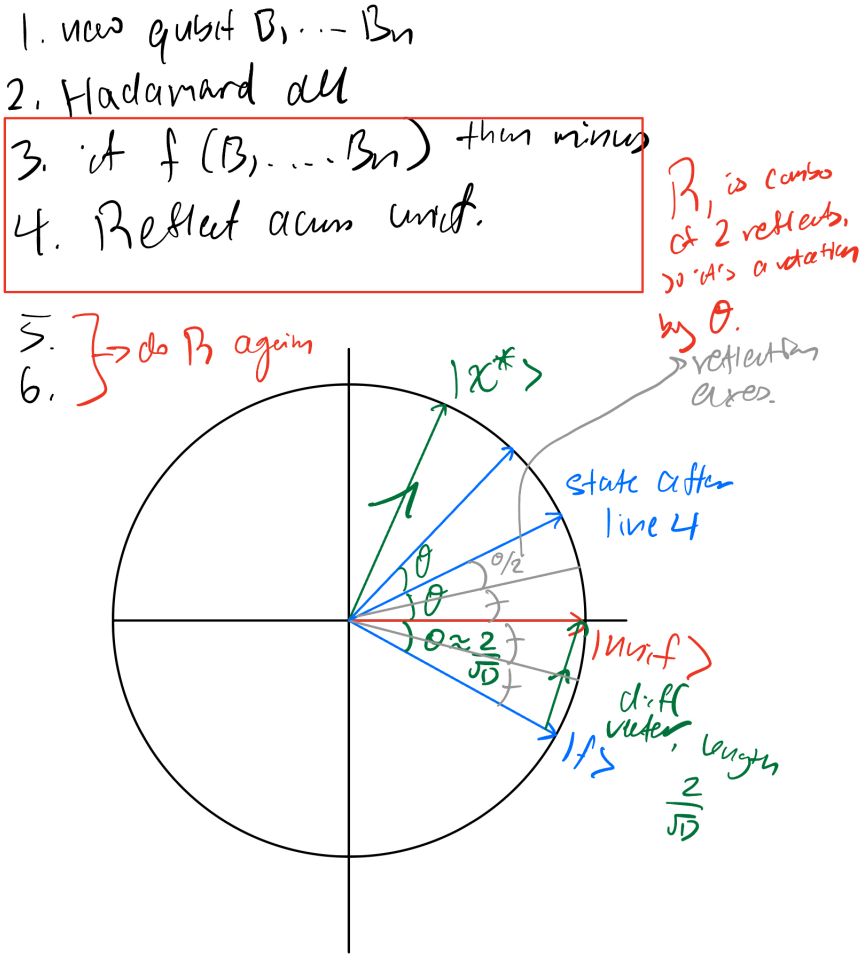
54.2 Solutions

D. If you have qubits in state $|11\ldots1\rangle$, then the amplitude is minus'd. Otherwise it's a no-op.

55 Grover's Algorithm, Part 2

Video #055/100 Table of Contents

4. Grover's Idea: reflect across $|unif\rangle$. $O(n+m)$ quantum ops. From we can easily create the reflection as $|unif\rangle$ is easy to make (just Hadamard all). Then do if (OR on all qubits) then minus, then Hadamard all again.



Starting from $|unif\rangle$, repeatedly doing R, we rotate $\theta, 2\theta, 3\theta$ toward $|x^*\rangle$. Recall that θ is small, so the diagram is a bit misleading. To get to $|x^*\rangle$, we need to travel $90° - \frac{\theta}{2} \approx 90°$ since θ is so small (remember, $\theta \approx \frac{2}{\sqrt{D}} = \frac{2}{2^n} = \frac{1}{2^{n-1}}$). So the number of times we should repeat R is about $\frac{90°}{\theta} = \frac{90°}{\frac{2}{\sqrt{D}}} = \sqrt{D} \times 45° = \sqrt{D} \times \frac{\pi}{4}$, which is less than $\sqrt{D} = \sqrt{2^n}$. This is where we get the $O(\sqrt{2}^n)$ time complexity.

To recap: Grover's Algorithm

1. new qubit $B_1 \dots B_n$ — $O(n)$
2. Hadamard All — $O(n)$
3. For $t = 1 \dots \lfloor \frac{\pi}{4}\sqrt{2^n} \rfloor$: — $O(\sqrt{2^n})$ iterations
   $R(B_1 \dots B_n)$ — $O(n+m)$, since we run the code from C twice
4. Extract All — $O(n)$

Final time complexity of $O(\sqrt{2^n} \times (m+n)) \in O(\sqrt{2}^n)$ by definition of Big O.

This will print $|x^*\rangle$ with error probability less than $\frac{1}{2^n}$.

55.1 Exercises: 2-bit Grover

Suppose we are doing Grover's algorithm with $n = 2$. So $f : \{0,1\}^2 \to \{0,1\}$ is a Boolean function with 2 input bits and 1 output bit, and we are promised that among the 4 possible two-bit input strings, exactly one, x^*, has $f(x^*) = 1$.

Assume we have two qubits B_1, B_2 and we can make calls to if $f(B_1, B_2)$ then minus.

- What is $|unif\rangle$, the normalized uniform superposition on B_1, B_2?
- Suppose x^* happens to be "10". If we apply if $f(B_1, B_2)$ then minus to the state $|unif\rangle$ and call the result $|f\rangle$, what is $|f\rangle$?
- The vector $|unif\rangle - |f\rangle$ is equal to r times $|x^*\rangle$ for some scalar r. What is the precise numerical value of r?
- Suppose the state vector $|f\rangle$ is reflected across the vector $|unif\rangle$. Call the resulting state vector $|new\rangle$. What is the precise angle between $|new\rangle$ and $|x^*\rangle$, in radians?
- If we were to now perform "extract all" on the state $|new\rangle$, what would be the probability the outcome is x^*, i.e., "10"?

55.2 Solutions

- $|unif\rangle = \sqrt{\frac{1}{2^2}} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
- $|f\rangle = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$
- $r = 1$ as $|unif\rangle - |f\rangle = \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix} = |10\rangle = |x^*\rangle$, which has length $\frac{2}{\sqrt{D}} = \frac{2}{\sqrt{2^2}} = \frac{2}{2} = 1$.
- Recall from the exercises for Video #052/100 that reflecting $|f\rangle$ across $|unif\rangle$ is given by the matrix $\mathbb{1} - 2\,|unif\rangle\langle unif|$ times $|f\rangle$. Note $\langle unif|f\rangle = \frac{1}{2}(1,1,1,1) \cdot \frac{1}{2}(1,1,-1,1) = \frac{1}{4}(1+1-1+1) = \frac{1}{2}$.

  $$|new\rangle = |f\rangle - 2\,|unif\rangle\langle unif|f\rangle = \frac{1}{2}\big((1,1,-1,1) - (1,1,1,1)\big) = \frac{1}{2}(0,0,2,0) = -|10\rangle$$

  So the angle between $|new\rangle$ and $|10\rangle$ is π radians.
- $\mathbb{P}(|x*\rangle) = \mathbb{P}(|10\rangle) = \langle new|10\rangle^2 = \langle -10|10\rangle^2 = (-1)^2 = 1$. We are guaranteed to see x^*, AKA "10".

56 Grover's SAT Speedup is Unimprovable (Lecture 15)

Video #056/100 Table of Contents

Recap: Given classical AND/OR/NOT code C, computing $f : \{0,1\}^n \to \{0,1\}$ with unique $x^* \in \{0,1\}^n$ such that $f(x^*) = 1$. Goal: output x^*.

Gover's Algorithm:

- Convert C to quantum op "if f then minus"
- Prepare $|unif\rangle = \frac{1}{D}\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}$ where $D = 2^n$, and also quantum operation "reflect across $|unif\rangle$"
- Let R be the combo of the previous 2 operations
- Within the 2-D plane containing $|unif\rangle$ and $|x^*\rangle$, R rotates state vectors by $\theta \approx \frac{2}{\sqrt{D}}$.
- Start from $|unif\rangle$, repeat $R \approx \frac{\frac{\pi}{2}}{\theta}$ times, get (almost exactly) to $|x^*\rangle$. Then "extract all".

This solves (unique)-SAT-(search) with less than roughly $\sqrt{D} = \sqrt{2^n} \approx 1.4^n$ operations!

Could there be an even faster quantum SAT algorithm? Probably not.
Why? Recall "SETH": classical algorithms for SAT can't beat brute force 2^n. If you don't get code C, but merely a "black box" for f, then you need 2^n calls to the box to find x^*. Until you try all possibilties you might miss it.
Quantum setting: what if you just have a "black box" for if f then minus?

Theorem (1994): In this setting, you must make $\geq \sqrt{2}^n$ calls to the box to find x^* (with high probability).

We still believe that quantum computers require exponential time to solve SAT (or any NP-complete problem).

56.1 Exercises

Consider the following form of the SAT problem: Given AND/OR/NOT code C with n variables and $m \leq O(n^2)$ lines, is there any satisfying string?

True or false: It is known that, in principle, SAT can be solved by classical computers in at most $O(1.5^n)$ time.

True or false: It is known that, in principle, SAT can be solved by quantum computers in at most $O(1.5^n)$ time.

True or false: According to Prof. O'Donnell, it is generally believed that SAT can be solved by quantum computers in at most $O(n^{15})$ time.

56.2 Solutions

False, True, False.

57 Grovering with 's' satisfying strings

Video #057/100 Table of Contents

How do we relax the promise that there is a "unique" x^* satisfying string?

Suppose we (somehow) knew there were exactly 3 x_1^*, x_2^*, x_3^* satisfying C. Goal: find one of them.

$|unif\rangle = \frac{1}{\sqrt{D}} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$, where $D = 2^n$.

if f then minus: $|unif\rangle \mapsto |f\rangle = \frac{1}{\sqrt{D}}(1, 1, \ldots, 1, -1, 1 \ldots, 1, -1, 1, \ldots, 1, -1, 1, \ldots, 1)$
$\implies |unif\rangle - |f\rangle = \frac{2}{\sqrt{D}}(|x_1^*\rangle + |x_2^*\rangle + |x_3^*\rangle)$ Define: $|goal\rangle = \frac{1}{\sqrt{3}}\left(|x_1^*\rangle + |x_2^*\rangle + |x_3^*\rangle\right)$. goal is a unit vector and a good goal: if we get to this state and then extract all, we're guaranteed to get a satisfying string.

- $|unif\rangle - |f\rangle = 2 \cdot \sqrt{\frac{3}{D}} \cdot |goal\rangle$
- difference vector has length $2 \cdot \sqrt{\frac{3}{D}}$
- $|goal\rangle$ is in the 2-D plane of $|unif\rangle$, $|f\rangle$.

Basically same setup as Grover, but $|x^*\rangle$ is now $|goal\rangle$ and the angle between $|unif\rangle$ and $|f\rangle$ is now $\theta \approx 2 \cdot \sqrt{\frac{3}{D}}$. We can swap "3" out with the number of satisfying strings (before it was just 1, so for Grover's, $\theta \approx \frac{2}{\sqrt{D}}$). If we make the same R as before, if we start at $|unif\rangle$, and repeat R approximately $\frac{\frac{\pi}{2}}{\theta}$ times which is $\frac{\pi}{4}\sqrt{\frac{2^n}{3}}$. We'll get *extremely close* to $|goal\rangle$. This is faster than Grover's by a factor of $\sqrt{3}$. Notice that, if we have 2^n satisfiying strings, we can repeat R approximately $\frac{\pi}{4}\sqrt{\frac{2^n}{2^n}} = \frac{\pi}{4}$ which rounds down to 0 times. Obviously, if we know there's 2^n satisfying strings, what's even the point of Grover's!

57.1 Exercises

Suppose C is some classical AND/OR/NOT code computing $f : \{0,1\}^{100} \to \{0,1\}$. As in Grover's algorithm, you create the subroutine "R" that does "if f then minus" and then "reflect across unif".

Suppose you (somehow) know that exactly 1 of the 2^{100} possible input strings for f is satisfying, and you'd like to find it. You plan to start from $|unif\rangle$, apply "R" many times, and then "extract all" in the hopes of getting the satisfying string. Which of the following is closest to the ideal number of times to apply "R"?

Same question, but this time you (somehow) know that exactly 2^{40} out of the 2^{100} possible inputs are satisfying. Now which of the following is closest to the ideal number of times to apply "R"?

Choices are all multiples of 10 from 10 to 100, or of the form 2^n, where n is a multiple of 10 from 10 to 100.

57.2 Solutions

Rotate $\frac{\frac{\pi}{2}}{\theta} \approx \frac{\frac{\pi}{2}}{\frac{2}{\sqrt{D}}} = \frac{\pi}{4}\sqrt{D} = \frac{\pi}{4}2^{50}$ So the closest is 2^{50}.

Rotate $\frac{\frac{\pi}{2}}{\theta} \approx \frac{\frac{\pi}{2}}{2\sqrt{\frac{2^{40}}{D}}} = \frac{\pi}{4}\sqrt{2^{60}} = \frac{\pi}{4}2^{30}$ So the closest is 2^{30}.

58 Tiny angle errors are not a problem

Video #058/100 Table of Contents

If we do "extract all" on $\left|\tilde{goal}\right\rangle$, are we still likely to see x_1^*, x_2^*, x_3^*? Let a_1, a_2, a_3 be amplitudes of $|tildegoal\rangle$ on x_1^*, x_2^*, x_3^*. Then $\mathbb{P}(\text{output satisfying string}) = a_1^2 = a_2^2 + a_3^2 = \left\|(a_1, a_2, a_3)\right\|^2$.

Note $\varepsilon^2 = length(|err\rangle)^2 =$ sum of squares of differences in amplitudes between $|goal\rangle$ and $\left|\tilde{goal}\right\rangle \geq \left(\sqrt{\frac{1}{3}} - a_1\right)^2 + \left(\sqrt{\frac{1}{3}} - a_2\right)^2 + \left(\sqrt{\frac{1}{3}} - a_3\right)^2 = \left\|(\sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}}) - (a_1, a_2, a_3)\right\|^2 \leq \varepsilon$

Worst case: (a_1, a_2, a_3) is a length $(1-\varepsilon)$ vector in the direction of $(\sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}})$. So it's squared length is $(1-\varepsilon)^2 = 1 - 2\varepsilon + \varepsilon^2 \geq 1 - 2\varepsilon$. Then $\mathbb{P}(\text{output satisfying string}) \geq 1 - 2\varepsilon$

So the probability of the output not being a satisfying string is $1 - \mathbb{P}(\text{output satisfying string}) \leq 2\varepsilon \leq 2\sqrt{\frac{3}{D}}$.

Lesson: if you get your state vector almost right, that's good enough.

58.1 Exercises

Definition: $|err\rangle = |goal\rangle - \left|\tilde{goal}\right\rangle$ Write $\varepsilon = length(|err\rangle)$. Check: $\varepsilon \leq 2 \cdot \sqrt{\frac{3}{D}}$.

58.2 Solutions

$\left|\tilde{goal}\right\rangle$ is at most angle θ from $|goal\rangle$. If it were further, then we've either done one too few or one too many R's (each R is a rotation by θ). Then at worst, $|err\rangle$ has length $|\sin\theta|$. But we know $|\sin x| \leq x$ for all x, so at worst, $|err\rangle$ has length θ, and we already said θ is at most $2 \cdot \sqrt{\frac{3}{D}}$. So at worst, $|err\rangle$ has length at most $2 \cdot \sqrt{\frac{3}{D}}$.

59 Grover when you know 'p' to within 1%

Video #059/100 Table of Contents

What happens when you don't know how many satisfying strings you have? In general, suppose you (somehow) know there exists exactly s satisfying strings for C. Define $p = \frac{s}{2^n}$ = fraction of strings that are satisfying. So $|unif\rangle - |f\rangle$ has length $2\sqrt{p}$. If p is small, then $\theta \approx 2\sqrt{p}$. So we do about $\frac{\pi}{4}\sqrt{\frac{2^n}{s}} = \frac{\pi}{4}\sqrt{\frac{1}{p}}$ repetitions of R.

Quantum algorithms hallmark: "square-root savings" compared with classical algorithms: Brute force algo: pick random strings and plug into C until you get a satisfying one. This takes about $\frac{1}{p}$ trials. Compare that to $O(\sqrt{\frac{1}{p}})$ trials.

Suppose (somehow) you knew p to within 1%, i.e. you (alg) know some # $\hat{p}$ such that $.99p \leq \hat{p} \leq 1.01p$. Just define $\hat{\theta} = 2\sqrt{\hat{p}}$. This $\hat{\theta}$ is within 1% of "ideal" $\theta = 2\sqrt{p}$. So if the alg repeats R $\frac{\frac{\pi}{2}}{\hat{\theta}}$ times, the final state's angle from $|goal\rangle$ is within 1% of 90° in the 2-D picture.

Then measuring at the end gives a satisfying string with error probability less than 1%.

# of operations: ran R $\frac{\frac{\pi}{2}}{\hat{\theta}} \approx \frac{\pi}{4}\sqrt{\frac{1}{p}}$ times since $\hat{\theta}$ is within 1% of θ.

59.1 Exercises: Bias manipulation

Let C be some classical 1-output-bit AND/OR/NOT code. (So it takes as input a certain fixed number of bits, and outputs 1 bit.) We will define $p(C)$ to be the *fraction* of all possible input strings x for C that are "satisfying". In other words, if C takes in n input bits, then

$$p(C) = \frac{\#\{x \in \{0,1\}^n : C(x) = 1\}}{2^n}$$

The best way to think about this is that if you were to input a completely uniformly random input string to C, the output bit would be like a $p(C)$-biased coin: it comes out "1" with probability $p(C)$ and comes out "0" with probability $1 - p(C)$. (probability enthusiast would call it a Bernoulli(p) random variable)

This problem is about some little tricks you could do, even if you don't understand how C works, to manipulate the "bias of the coin".

(a) Describe a simple algorithm that takes as input some code C as above, and outputs some other 1-output-bit AND/OR/NOT code C' such that $p(C') = 1 - p(C)$. Please note that your simple algorithm is like a text-processing algorithm, and in particular there's no easy way for it to "know" what $p(C)$ is, given C.

formally, "simple" means that if the input is m lines of code, the your algorithm should take poly(m) time; probably at worst $O(m \log m)$ time in fact. But I don't mean for your to get into actual time complexity or anything; I'm sure we will all be able to agree on what is "simple" and what isn't.

(b) Same question, but with $p(C') = \frac{p(C)}{2}$.

(c) Same question, but with $p(C') = \frac{p(C)}{4}$.

(d) Same question, but with $p(C') = p(C)^2$.

(e) Same question, but with $p(C') = 1 - (1 - p(C))^{10}$.

59.2 Solutions

Define $C^\neg$ as taking C and adding one NOT line at the end to negate the output. So for all $x \in \{0,1\}^n$, $C(x) = 1 \iff C^\neg(x) = 0$, and $C(x) = 0 \iff C^\neg(x) = 1$.

(a) Set $C' = C^\neg$. That way, every string that didn't satistfy C (AKA $C(x) = 0$) becomes a satisfying string, and every string that did satisfy C becomes a non-satisfying string. So assuming C had s satisfying strings, it now as s non-satisfying strings (the remaining $2^n - s$ strings are now satisfying).

$\implies p(C') = \frac{2^n - s}{2^n} = 1 - p(C)$, given there were s satisfying strings originally for C for n input bits.

(b) Define $C'(x) : \{0,1\}^{n+1} \to \{0,1\}$ via $C(x_1x_2 \dots x_n y) = 1 \iff C(x_1x_2 \dots x_n) = 1 \wedge y = 1$.

To make C', you just take C, copy it. Also add another bit y. Now C' takes in bitstrings of length $n+1$, of the form x augment y, where x is a bitstring of length n. if $C(x) = 1$ and $y = 1$, $C'(x \text{ augment } y) = 1$. else 0. This way, there are still s bitstrings that satisfy C' (they're the same as the s bitstrings that satisfy C, but with a 1 augmented to the end), but now there are 2^{n+1} total possible bitstring inputs. $p(C') = \frac{s}{2^{n+1}} = \frac{\frac{s}{2^n}}{2} = \frac{p(C)}{2}$

(c) Define $C'(x) : \{0,1\}^{n+2} \to \{0,1\}$ via $C(x_1x_2 \dots x_n y_1 y_2) = 1 \iff C(x_1x_2 \dots x_n) = 1 \wedge y_1y_2 = 11$.

Similar idea to part (b). Just augment each bitstring with two bits y_1y_2. Now C' takes in bitstrings of length $(n+2)$, of the form x augment y_1y_2, where x is a bitstring of length n. If $C(x) = 1$ and $y_1y_2 = 11$, $C'(x \text{ augment } y_1y_2) = 1$. else 0. This way, there are still s bitstrings that satisfy C' (they're the same as the s bitstrings that satisfy C, with "11" augmented to the end), but now there are 2^{n+2} total possible bitstring inputs. $p(C') = \frac{s}{2^{n+2}} = \frac{\frac{s}{2^n}}{2^2} = \frac{p(C)}{4}$

(d) Define $C'(x) : \{0,1\}^{2n} \to \{0,1\}$ via $C(x_1x_2 \dots x_n y_1 y_2 \dots y_n) = 1 \iff C(x_1x_2 \dots x_n) = 1 \wedge C(y_1y_2 \dots y_n) = 1$.

To make C', take C. But make C' take in bitstrings of length $2n$. $C'(x) = 1$ iff the first n bits, which we'll call x, satisfy $C(x) = 1$, and the second n bits, which we'll call y, satisfy $C(y) = 1$. Then there's s^2 bitstrings that satisfy C' (pick one of the s to be x, then pick one of the s to be y, with replacement), but now there's $(2^n)^2 = 2^{2n}$ total bitstrings. $p(C') = \frac{s^2}{2^{2n}} = \left(\frac{s}{2^n}\right)^2 = p(C)^2$

(e) Define $C'(x) : \{0,1\}^{10n} \to \{0,1\}$ via $C(a_1a_2 \dots a_n b_1b_2 \dots b_n \dots j_1j_2 \dots j_n) = 1 \iff C^\neg(a_1a_2 \dots a_n) = 1 \wedge C^\neg(b_1b_2 \dots b_n) \wedge \dots \wedge C^\neg(j_1j_2 \dots j_n) = 1$ (and 0 otherwise). Then at the very end, we toggle the output in C'. So actually, $C(a_1a_2 \dots a_n b_1b_2 \dots b_n \dots j_1j_2 \dots j_n) = 1 \iff C(a_1a_2 \dots a_n) = 1 \vee C(b_1b_2 \dots b_n) = 1 \vee \dots \vee C(j_1j_2 \dots j_n) = 1$.

Now we combine parts (a) and (d). Take C and negate the output. Now make C' take in bitstrings of length $10n$; each bitstring x is split into $x_1, x_2, x_3, \dots, x_{10}$. Now C' outputs 1 if and only if all x_i satisfy $C(x_i) = 0$. Now toggle the output.

60 1-Qubit Rotation Estimation: Overview (Lecture 16)

Video #060/100 Table of Contents

Recall: Suppose C is m lines of AND/OR/NOT code computing $f : \{0,1\}^n \to \{0,1\}$ and a p fraction of $x \in \{0,1\}^n$ are satisfying. From C, can form quantum subroutine R ($O(m+n)$ operations) which rotates the 2-D plane of $|unif\rangle$ and $|goal\rangle$ by $\theta \approx 2\sqrt{p}$ is the angle between $|unif\rangle$ and $|f\rangle$. (Precisely, $\cos\theta = \langle f|unif\rangle = 1-2p$). Last time, if you (somehow) knew p or θ to within 1%, you can (with high probability) find a satisfying string x using $\leq \sqrt{\frac{1}{p}}$ calls to R.

We will show that you can find θ to within 1% in order $\sqrt{\frac{1}{p}}$ work.

Rotation Estimation (vague dsecrption): given access to a state vector $|start\rangle$ and a subroutine MysteryRotation which rotates $|start\rangle$ by a mystery angle θ, estimate θ. For SAT, okay so assume $p \leq \frac{1}{16} \implies \theta < 30°$.

$$Proof: \theta \approx 2\sqrt{p} \leq 2\sqrt{\frac{1}{16}} = 2 \times \frac{1}{4} = \frac{1}{2} \leq \frac{\pi}{6} = 30°(\text{since } \pi > 3)$$

Simplest Rotation Estimation Problem Setup

- MysteryRotation is a 1-qubit operation
- Assume $\theta < 30°$ and $\theta > 0°$
- goal: output estimate $\hat{\theta}$ with relative error $\leq 1\%$

  $0.99\theta \leq \hat{\theta} \leq 1.01\theta$

Theorem: this is doable (with high probability) using $O(\frac{1}{\theta})$ calls to MysteryRotation (without knowing anything about θ).

60.1 Exercises: an amplitude-centric view of Grover.

(a) Let $|u\rangle$ be a unit vector and let $|v\rangle$ be any vector. We can write $|v\rangle = m \cdot |u\rangle + |w\rangle$, where m is a scalar and $|w\rangle$ is perpendicular to $|u\rangle$, (i.e. $\langle u|w\rangle = 0$). What is a formula for m and for $|w\rangle$ in terms of $|u\rangle$, $|v\rangle$? Why not draw a little 2-D picture?

(b) Suppose

$$|unif\rangle = \sqrt{\frac{1}{8}}\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix} \quad \text{and} \quad |Y\rangle = \sqrt{\frac{1}{8}}\begin{bmatrix}y_1\\y_2\\y_3\\y_4\\y_5\\y_6\\y_7\\y_8\end{bmatrix}$$

where $y_1, \ldots, y_8$ are real numbers. Compute $\langle unif|unif\rangle$ and $\langle unif|Y\rangle$, and describe each of these results in a couple of English words.

(c) If we express $|Y\rangle$ as $m \cdot |unif\rangle + |\Delta\rangle$, where m is a scalar and $|\Delta\rangle$ is perpendicular to $|unif\rangle$, then what is m and what is $|\Delta\rangle$?

(d) In terms of the preceding objects, write a formula for "$|Y\rangle$ reflected across $|unif\rangle$".

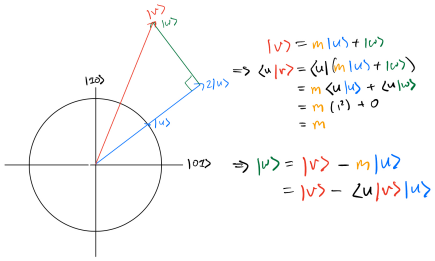
(e) Suppose $y_1 = y_2 = \cdots = y_8 = 1$, and then we apply the following transformations to $|Y\rangle$:

1. Negate the first coordinate
2. Reflect across $|unif\rangle$
3. Negate the first coordinate
4. Reflect across $|unif\rangle$

Work out the resulting entries of $|Y\rangle$ after each of the above 4 steps. For 3-qubit Grover, now would be a good time to measure...

60.2 Solutions

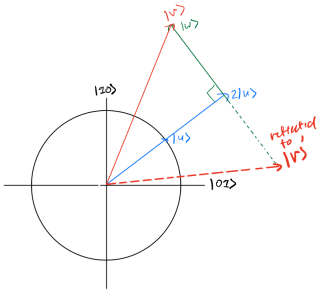
(a) $m = \langle u|v\rangle$ and $|w\rangle = |v\rangle - \langle u|v\rangle\,|u\rangle$



(b) $\langle unif|unif\rangle = \big\||unif\rangle\big\|^2 = (1)^2 = 1$. $\langle unif|Y\rangle = \frac{1}{8}(y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8)$
$\langle unif|unif\rangle$ is 1, and $\langle unif|Y\rangle$ is the average value of $y_1, \ldots, y_8$.

(c) Use what we did in part (a). Set $u = |unif\rangle$ and $v = |Y\rangle$. Then $m = \langle unif|Y\rangle$ and $|\Delta\rangle = |Y\rangle - \langle unif|Y\rangle\,|unif\rangle = |Y\rangle - m\,|unif\rangle$.

(d) $|Y\rangle$ reflected across $|unif\rangle$ is $|Y\rangle - 2\,|\Delta\rangle$.



(e) From part (c), we can deduce reflection across $|unif\rangle$ is $|Y\rangle \mapsto -|Y\rangle + 2m\,|unif\rangle = 2m\,|unif\rangle - |Y\rangle$.

$|Y\rangle \mapsto \sqrt{\frac{1}{8}}(-1, 1, \ldots, 1)$ (line 1)

$m = \frac{7(1)+1(-1)}{8} = \frac{6}{8} = \frac{3}{4} \implies 2m = \frac{3}{2}$.

Then $|Y\rangle \mapsto \sqrt{\frac{1}{8}}(\frac{3}{2}, \frac{3}{2}, \ldots, \frac{3}{2}) - \sqrt{\frac{1}{8}}(-1, 1, \ldots, 1) = \sqrt{\frac{1}{8}}(\frac{5}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ (line 2)

$|Y\rangle \mapsto \sqrt{\frac{1}{8}}(-\frac{5}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ (line 3)

$m = \frac{7(\frac{1}{2})+1(-\frac{5}{2}))}{8} = \frac{\frac{2}{2}}{8} = \frac{1}{8} \implies 2m = \frac{1}{4}$.

Then $|Y\rangle \mapsto \sqrt{\frac{1}{8}}(\frac{1}{4}, \frac{1}{4}, \ldots, \frac{1}{4}) - \sqrt{\frac{1}{8}}(-\frac{5}{2}, \frac{1}{2}, \ldots, \frac{1}{2}) = \sqrt{\frac{1}{8}}(\frac{11}{4}, -\frac{1}{4}, \ldots, -\frac{1}{4})$. (line 4)

61 Detecting "medium" 1-qubit rotations

Video #061/100 Table of Contents

Key Idea: If θ happens to be a "medium" angle, meaning $30° \leq \theta \leq 60°$, then it's easy to notice this.

`IsMedium(MysteryRotation)`

1. count := 0
2. for $t = 1 \ldots 100$:

   new qubit A

   MysteryRotation on A

   extract all

   if outcome = "1", count ++
3. if $20 \leq$ count ≤ 80, return "yes", else "no".

Properties of this algorithm:

- If θ is "medium", then IsMedium returns "yes" with high probability ($> 90\%$). Why? in one measurement MysteryRotation on A, $\mathbb{P}(\text{outcome} = \text{"1"}) = \sin^2\theta$. Since θ is "medium", $30° \leq \theta \leq 60° \implies \frac{1}{4} \leq \sin^2\theta \leq \frac{3}{4}$. So we expect count to be between 25 and 75, so the probability that count is between 20 and 80 is high ($> 90\%$).
- If θ is not "mediumISH", then IsMedium returns "no" with high probability ($> 90\%$). Where "mediumISH" means $22.5° \leq \theta \leq 67.5°$. We use mediumISH here because it's difficult to distinguish between incredibly close angles. However, these relaxed bounds for mediumISH are true for high probability. Why? not mediumISH means either $\theta < 22.5°$ or $\theta > 67.5°$. Then each measurement has $\mathbb{P}(\text{"1"})$ either < 0.147 or > 0.853, so we expect count < 15 or > 85, so less than 20% chance count is in 20 to 80.

So, we can "reliably" distinguish case "θ is medium" and case "θ is not even mediumISH". Temporarily, pretend that "reliable" means 100% reliable. Then IsMedium(MysteryRotation) says "no" if and only if θ is for sure not medium, and "yes" if and only if θ is at least mediumISH. This works because we can just change the number of iterations in IsMedium and change our count thresholds to get to arbitrary accuracy.

61.1 Exercises

Suppose you are promised to be in either Case 1: $30° < \theta \leq 60°$ or Case 2: $0° < \theta \leq 30°$. True or false: you can reliably determine which of the two cases you're in with a reasonably small number of calls to MysteryRotation? (Say that reliably means with correctness probability $> 90\%$, and reasonable small means less than 1,000,000.)

Suppose you are promised to be in either Case 1: $80° < \theta \leq 90°$ or Case 2: $0° < \theta \leq 45°$. True or false: you can reliably determine which of the two cases you're in with a reasonably small number of calls to MysteryRotation?

61.2 Solutions

False, True. In the first scenario, too hard to distinguish between really close angles like 29.9° (Case 2) and 30.1° (Case 1). Second scenario is doable because the cases are further apart.

62 1-qubit Rotation Estimation to "factor 2" error

Video #062/100 Table of Contents

Now we do the binary search component.

`Factor2Estimate(MysteryRot)`

1. For $k = 2, 4, 8, 16, \ldots$

   if IsMedium(MysteryRotation^k return "$\hat{\theta} = \frac{45^\circ}{k} = \frac{\pi}{4k}$")

   (MysteryRotation^k is running MysteryRotation k times consecutively, same as rotating by $k\theta$)

Analysis of Factor2Estimate

- Correctness (if the alg returns, it returns the correct answer, AKA $\hat{\theta}$ is within a factor of 2 of θ)

  If Factor2Estimate returns, then it got "yes" on some call to IsMedium(MysteryRotation^k). Then $k\theta$ is for sure mediumISH by correctness of IsMedium (and assumption of 100% reliability). Then $22.5^\circ \leq k\theta \leq 67.5^\circ$, which is within factor 2 of 45°, which means $\hat{\theta} = \frac{45^\circ}{k}$ is wthin factor 2 of θ.

- Termination:

  There's a unique power of 2, k^*, such that $k^*\theta$ is medium. Once Factor2Estimate reaches this k^*, IsMedium for sure says yes (due to our reliability assumption). So Factor2Estimate will return.

- if θ is really small, then Factor2Estimate takes a long time to run.

- We know that by the k^* iteration, $k^*\theta \leq 60^\circ$, so $k^* \leq \frac{60^\circ}{\theta}$.

  IsMedium calls MysteryRotation $t = 100$ times, and MysteryRot^k calls MysteryRotation k times, so we'll get $100 \times 1 + 100 \times 2 + 100 \times 4 + \cdots + 100 \times k^* = 100 * 2k^* \in O(k^*)$ calls to MysteryRotation.

We've basically proven our Theorem for Rotation Estimation Video #060/100, except now we're close to a factor 2 of θ, not 1%. Also we pretended IsMedium is 100% reliable.

62.1 Exercises

Write a computer program that simulates flipping a p-biased coin a hundred million (10^8) times, where $p = 0.2$. That is, the coin comes up "1" with probability 0.2 and "0" with probability 0.8. Let $\hat{p}$ denote the fraction of times your coin came up "1". Have your program print out $\hat{p}$. Actually, run your program 20 times, thereby getting 20 different "estimates" $\hat{p}$. Did you ever get a $\hat{p}$ that didn't start with .2000 or .1999? If not, how close did it come? Did you ever get something bigger than .20008 or smaller than .19992?

62.2 Solutions

notes.tex 9+ 062.py U

15-459 > sim > 062.py > ...

```python
import numpy as np

def flip_coin(p, n=1):
    """
    Simulates flipping a coin with probability `p` of landing heads.

    Parameters:
    - p (float): Probability of the coin landing heads (0 <= p <= 1).
    - n (int): Number of flips to simulate (default is 1).

    Returns:
    - list: A list of outcomes where True represents heads and False represents tails.
    """
    if not (0 <= p <= 1):
        raise ValueError("Probability p must be between 0 and 1.")

    # Simulate n flips with the given probability
    flips = np.random.rand(n) < p

    return flips

# Example usage
if __name__ == "__main__":
    p = 0.2  # Probability of 1
    n = 10**8   # Number of coin flips
    for i in range(20):
        results = flip_coin(p, n)
        print(f"phat_{i+1} = {np.sum(results) / 10**8}")
```

OUTPUT PORTS TERMINAL PROBLEMS 19 DEBUG CONSOLE

```
(15-xxx) tyleryang@Mac:~/Developer/School/Self-Study/15-459/sim$ python3 062.py
phat_1 = 0.20005395
phat_2 = 0.19999827
phat_3 = 0.19995335
phat_4 = 0.20007212
phat_5 = 0.19995396
phat_6 = 0.19996891
phat_7 = 0.20004084
phat_8 = 0.19999483
phat_9 = 0.20002534
phat_10 = 0.19999099
phat_11 = 0.19995155
phat_12 = 0.19997966
phat_13 = 0.19994837
phat_14 = 0.20001577
phat_15 = 0.20004594
phat_16 = 0.20003855
phat_17 = 0.20002705
phat_18 = 0.20004702
phat_19 = 0.20004574
phat_20 = 0.19997317
(15-xxx) tyleryang@Mac:~/Developer/School/Self-Study/15-459/sim$
```

No, did not get any result that did not start with a 0.2000 or 0.1999. Also did not get something bigger than 0.20008 or smaller than 0.19992.

63 Factor-1% Rotation Estimation loose ends

Video #063/100 Table of Contents

What about $\theta = 0$? Then our loop would run for $\frac{1}{0} = \infty$ iterations, so it would fail. Maybe we'd stop at a k_{max}, upon reaching you'd return "θ is $\frac{1}{k_{max}}$"

If $\theta < \frac{0.25}{k_{max}}$, the algorithm will return this. If it ever comes to this case, the number of MysteryRotation calls is $O(k_{max})$. If we go back to Grover SAT scenario, we'd do this with $k_{max} = \sqrt{2^n}$, since $\frac{1}{p}$ was either $\frac{1}{2^n}$ or 0 for the Unique-Circuit-SAT case. So if we found $\theta < \frac{1}{\sqrt{2^n}}$, then we'd know $p = 0$.

The fix for the reliability assumption that we pretended was true is a well-known fix for any probabilistic algorithm: repeat it a bunch of times.

Idea: in Factor2Estimate, we'd call IsMedium 3 times. Then we'll just take the majority answer. Then the proabability if getting it wrong is the probability all 3 IsMediums are wrong or 2 IsMediums are wrong (10% = 0.1 each wrong) which is $\binom{3}{2}0.1^2 0.9 + 0.1^3 = 0.028 < 2^{-3}$. Now we have probability of failure that is 3%.

Fact: (Chernoff Bounds): repeat $O(t)$ times and take the majority answer. The failure probability becomes less than 2^{-t}. Definition: say a probabilistic algorithm works "with overwhelming probability" (*wowp*) if it has this feature. Special license for this class: whenever we have algorithm that works *wowp*, we'll just assume it's actually 100% reliable (and thus deterministic), with only a constant factor slowdown.

Sketching what will happen for bringing factor 2 error from θ to 1%. After Factor2Estimate, we know k such that MysteryRotation^k has rotation angle between 22.5° and 67.5°. We know $\theta' = k\theta \approx 45°$, up to a factor of 2. It suffices now to figure out θ' to within 1%. In essence, we've already shown $\frac{\theta'}{2} \leq est' \leq 2\theta'$, then we can divide by k to get a good estimate of θ.

63.1 Exercises

Suppose you're given access to a MysteryRotation instruction, operating on one qubit, promised to rotate by some angle between 0 and 30 degrees. You don't know it, but secretly $\theta = 2^{-400}$ radians. If you run the Factor2Estimate algorithm from lecture to estimate θ's value, how many times will it call MysteryRotation (wowp)? Select the closest answer:

20, 100, 200, 400, or 2 to the one of the previous numbers.

63.2 Solutions

Each IsMedium has 100 calls, and we have $k^* \leq \frac{60°}{2^{-400}} \approx 2^{400}$. So we need about $O(k^*)$ calls, and the only answer choice in that is 2^{400}.

64 Rotation Estimation with additive error (Lecture 17)

Video #064/100 Table of Contents

Recap: 1-qubit relative error for Rotation Estimation.

- MysteryRotation is a 1-qubit rotation by unknown θ between 0 and 30 degrees.
- With $O(\frac{1}{\theta})$ calls to MysteryRotation, can (*wowp*) estimate θ to within factor 2.
- At the end of Factor2Estimate, we know k such that MysteryRotation^k has rotation angle $k\theta$ between 22.5 and 67.5 degrees. So 45 degrees is within factor 2 of $k\theta$, which means $\frac{45^\circ}{k}$ is within factor 2 of theta.

Let's estimate $k\theta$ to within 1%. It's sufficient to estimate $k\theta$ to within $\pm.004$ radians. Why? $.004 \leq 1\% k\theta$ because $.004 \leq 1\% \cdot 22.5^\circ = 0.001 \cdot 0.4rad = .004$.

Theorem: (simple additive error Rotation Estimation) Given access to 1-qubit MysteryRotation' acting as rotation by θ' between 0 and 90 degrees, we can find an estimate of θ' within ± 0.004 using $O(1)$ calls to MysteryRotation' (*wowp*).

MysteryRotation' = MysteryRotation^k as $\theta' = k\theta$. So we have $O(1) \cdot k$ calls to MysteryRotation. But $k \propto \frac{1}{\theta}$, so we have $O(\frac{1}{\theta})$ calls to MysteryRotation.

Simple setting: "known 2-D plane" setup where MysteryRotation acts on span$\{|0\rangle, |1\rangle\}$.

Grover's setting is a semi-known 2-D plane, where we know $|unif\rangle$ in this 2-D plane we are analyzing, but we don't know $|goal\rangle$. So we know some stuff but not everything.

How did Rotation Estimation work? Make a new qubit in state 0, then MysteryRotation a bunch, then measure, and repeat. Can we do this in the Grover setting? Well in the simple setting, if the state vector at angle α from $|0\rangle$, we extract to see "1" with probability $\sin^2 \alpha$. Can we get a similar situation for the Grover setup? We need to get our state vector to some angle α AKA some power of 2 times θ away from $|unif\rangle$, then do a measurement that has probability of $\sin^2 \alpha$.

In Grover Setting: changes from the 1-qubit algorithm

- replace new qubit in $|0\rangle$ with preparing $|unif\rangle$
- replace extracting the 1 qubit and checking for outcome "1" (not "0") with the following subroutine
  1. Hadamard all (rotates space so $|unif\rangle$ goes back to all zeros $|00\ldots0\rangle$)
  2. extract all
  3. check if outcome is not $00\ldots0$.

  Why does this work? If state vector has angle α from $|unif\rangle$ before line 1, after line 1, the state vector is at angle α from $|00\ldots0\rangle$. So it's of the form $\cos\alpha\,|00\ldots0\rangle$ +other amplitudes on other states. So when you extract all, the probability of getting $|00\ldots0\rangle$ is $\cos^2\alpha$. so the probability you don't see all zeros is $1 - \cos^2\alpha = \sin^2\alpha$.

64.1 Exercises

Let C be some AND/OR/NOT code that has n input bits, k output bits, and m total instructions. (Assume for simplicity that $k, n \leq m$.) We can think of C's output as a nonnegative integer Y written in base 2.

Let's say that "running" C refers to choose its n input bits uniformly at random, evaluating C, and calling the output $\mathbf{Y}$. This $\mathbf{Y}$ is an integer valued random variable. Let μ be the expected value of $\mathbf{Y}$, and let σ^2 be the variance of $\mathbf{Y}$.

Suppose we run C consecutively T times (using fresh randomness each time), producing outputs $\mathbf{Y}_1, \ldots, \mathbf{Y}_T$. Then we define

$$\hat{\mu} = \frac{\mathbf{Y}_1 + \cdots + \mathbf{Y}_T}{T}$$

This random variable $\hat{\mu}$ is a reasonable estimate for μ.

(a) Compute $E[\hat{\mu}]$ and $Var[\hat{\mu}]$ in terms of μ and σ.

(b) As I hope you know, Chebyshev's inequality says that a random variable is within 3 standard deviations of its expected value with "high probability": specifically, with probability at least $\frac{8}{9} \approx 89\%$. Deduce that a classical randomized algorithm can, in $O(mT)$ time, get an estimate of μ that is within $\pm 3\frac{\sigma}{\sqrt{T}}$ of μ (with "high probability").

64.2 Solutions

(a) $E[\hat{\mu}] = E[\frac{\mathbf{Y}_1+\cdots+\mathbf{Y}_T}{T}] = \frac{1}{T}\left(E[\mathbf{Y}_1)] + \cdots + E[\mathbf{Y}_T]\right) = \frac{1}{T}(T\mu) = \mu$, and $Var[\hat{\mu}] = Var[\frac{\mathbf{Y}_1+\cdots+\mathbf{Y}_T}{T}] = \frac{1}{T^2}(Var[\mathbf{Y}_1] + \cdots + Var[\mathbf{Y}_T]) = \frac{1}{T^2}(T\sigma^2) = \frac{\sigma^2}{T}$.

(b) We estimate μ using $\hat{\mu}$, with expected value μ and standard deviation $\sqrt{Var[\hat{\mu}]} = \sqrt{\frac{\sigma^2}{T}} = \frac{\sigma}{\sqrt{T}}$. By Chebyshev's inequality, $\hat{\mu}$ is within 3 standard deviations of μ with high probabiliy, AKA $\hat{\mu}$ is within $\pm\frac{3\sigma}{\sqrt{T}}$ of μ with high probability. We get $\hat{\mu}$ by running C consecutively T times. Each run of C needs to run m lines, giving $O(mT)$ time overall. The desired result follows: in $O(mT)$ time, we get $\hat{\mu}$ to estimate μ within $\pm\frac{3\sigma}{\sqrt{T}}$ with high probability.

65 Rotation Estimation, n digits accuracy?

Video #065/100 Table of Contents

Moving on, we're going back to the important theorem. Recall $\tau = 2\pi$.

Theorem: (simple additive error Rotation Estimation) Given access to 1-qubit MysteryRotation acting as rotation by θ between 0 and $\frac{\tau}{4}$ degrees, we can find an estimate of θ within $\pm 0.003141\ldots$ (interval of width 0.001τ containing θ) using $O(1)$ calls to MysteryRotation (*wowp*).

`IntervalEstimate(MysteryRotation)` # assume θ is between 0 and $\frac{\tau}{4}$

1. count = 0
2. T := 120,000
3. for $i = 1 \ldots T$

   new qubit A

   MysteryRotation A

   extract all, if outcome is "1", count++
4. $\hat{q} := \frac{\text{count}}{T}$ $(q = \sin^2\theta)$
5. $\hat{\theta} = \arcsin\sqrt{\hat{q}}$
6. return interval "$\hat{\theta} \pm 0.0005\tau$" (width 0.001τ)

Remark: if you wanted an interval of width ε, you'd need $T \approx \frac{1}{\varepsilon^2}$ samples. But eventually, for the quantum factoring algorithm, we will want thousands of digits of accuracy ($\varepsilon = 10^{-1000}$). So this strategy of Rotation Estimation would take way too many measurements.

For n digits of accuracy, we need $10^{-n}\tau$ to be our interval width. So this would take $\frac{1}{(10^{-n})^2} = 100^n$ steps. But with quantum algorithms, we can get this down to 10^n steps. Still ridiculous, so how is this supposed quantum factoring algorithm going to work?

65.1 Exercises (continued from last one)

(c) Consider now the special case $k = 1$. Assume for simplicity that $\mu \leq \frac{1}{16}$. Show that $Var[\mathbf{Y}] \approx \mu$ (in the sense that, say, $.9\mu \leq Var[\mathbf{Y}] \leq \mu$)

65.2 Solutions

(c) $Var[\mathbf{Y}] = E[\mathbf{Y}^2] - \big(E[\mathbf{Y}]\big)^2 = E[\mathbf{Y}] - \mu^2 = \mu - \mu^2$
$\mu \leq \frac{1}{16}$
$\implies \mu^2 \leq \frac{1}{256} \approx 0.004$
$\implies 0.96\mu \leq \mu - \mu^2 \leq \mu$
$\implies 0.90\mu \leq \mu - \mu^2 \leq \mu$, as desired.

66 Factoring via Rotation Estimation: plan

Video #066/100 Table of Contents

Intuition/Sketch on how we're going to get 1000 digits of accuracy:

Suppose $\theta = 0.d_1d_2d_3d_4\ldots\tau$.

- Apply IntervalEstimate(MysteryRotation). This gives us d_1, d_2, d_3, maybe a little uncertainty on d_3.
- MysteryRotation^{10} rotates by 10θ which is $d_1.d_2d_3d_4\ldots\tau$, which is coterminal with $0.d_2d_3d_4\ldots\tau$. Now do IntervalEstimate on MysteryRotation^{10}, and we'll learn d_2, d_3, d_4, maybe a little uncertainty on d_4.
- Next: repeat for MysteryRotation^{100} to get d_3, d_4, d_5. and we keep going to keep as many digits of accuracy as necessary.
- We'll get up to the nth digit via IntervalEstimate(MysteryRotation$^{10^n}$) This uses about 10^n steps.

  Total # of calls to MysteryRotation:

  For this, we have IntervalEstimate times MysteryRotation^k which is

  $120,000 \cdot (1 + 10 + 100 + \cdots + 10^n) \in O(10^n)$
- But for each power k, we'll have a way to do IntervalEstimate(MysteryRotation$^{10^n}$) in n steps instead of 10^n. So in Factoring, we'll know how to implement MysteryRotation^k efficiently even if $k = 10^{1000}$. So we'll eventually get the time complexity down by quite a bit.

66.1 Exercises (continued from last one)

(d) Continuing the above $k = 1$ case, sketch an argument that there is a quantum algorithm using $O(mT)$ operations that gets an estimate of μ that is within $\pm\frac{3\sigma}{T}$ of μ (with high probability).

Remark: This T vs. $\sqrt{T}$ quantum improvement is a fairly general form of the "quantum quadratic speedup" for stats-type problems. The result in part (d), but for general $k > 1$, was proven in 2022.

66.2 Solutions

(d) As this video showed, if we want to estimate to 10^n digits of accuracy, AKA to $\varepsilon = 10^{-n}$, we can do so in 10^n steps. More generally, estimating to $\varepsilon = \frac{1}{T} = T^{-1}$ accuracy can be done so in $O(T)$ steps. But each step requires a run of C, so we multiply by m to get $O(mT)$ steps.

67 Rotation Estimation: n digits, 10^n steps

Video #067/100 Table of Contents

Let R be a 1-qubit rotation by angle θ between 0 and $\frac{\tau}{4}$. Say that secretly, $\theta = 0.1056029\ldots\tau$

Algorithm: first run IntervalEstimate(R) (wowp, should succeed). This returns an interval of width 0.001τ, like maybe the interval $[0.105\tau, 0.106\tau]$.
Now we know $\theta \in [0.105\tau, 0.106\tau]$

Let's think about R^{10}. It rotates by $10\theta = 1.056029\ldots\tau \equiv 0.056029\tau$
But using info from before, $\theta \in [0.105\tau, 0.106\tau] \implies 10\theta \in [1.05\tau, 1.06\tau] \equiv [0.05\tau, 0.06\tau]$. Now run IntervalEstimate(R^{10}), which returns an interval of width 0.001τ contaiing 10θ,
like maybe the interval $[0.0554\tau, 0.0564\tau]$
Therefore, $\theta \in [0.10554\tau, 0.10564\tau]$

Think about R^{100}. It rotates by $100\theta = 10.56029$. Then IntervalEstimate(R^{100}) might return another narrower interval of width 0.001τ, telling us 100θ in $[0.5597\tau, 0.5607\tau]$.

But for this step, 0.5540τ is bigger than $\frac{\tau}{4}$, so we had to pick some offset angle to move the angle back between 0 and 90 degrees (so we'd do IntervalEstimate(R_{offset})^{100}). Here, we'd pick offset as 180 degrees, or $\frac{\tau}{2} = 0.5\tau$.

67.1 Exercises

Suppose you are given some classical AND/OR/NOT code C computing a Boolean function $f : \{0,1\}^n \to \{0,1\}$. You may assume C has $O(n)$ instructions. You are promised that there are *exactly* three input strings $x_1^*, x_2^*, x_3^* \in \{0,1\}^n$ that satisfy C (i.e., make f output 1). Your goal is to find all three satisfying strings. Describe and analyze a method for doing this that succeeds with high probability and uses as most $O(n) \cdot \sqrt{2^n}$ time (i.e., this many total instructions). If you wish, you may state and use and true facts about Grover's algorithm that we learned in class. Your method will probably be "mostly quantum", but you can mix of classical and quantum operations if you feel you need to.

67.2 Solutions

Grover's has time complexity of $O(\sqrt{2^n} \times (m+n))$ (Video #055/100). Since C has $O(n)$ instructions, $m \in O(n)$, meaning Grover's here has $O(n\sqrt{2^n})$ time complexity. This is what the problem hints at, so we will be using Grover's.

Use Grover's for $s = 3$, AKA there are 3 satisfying strings. Each of the 3 satisfying strings has $\frac{1}{3}$ probability of being outputted. So now we repeat. We want high probability ($\geq 90\%$) that we see all 3 satisfying strings. In t trials,

$$\begin{aligned}
\mathbb{P}(\text{all strings appear}) &= 1 - \mathbb{P}(\geq 1 \text{ string is missing}) \\
&= 1 - \mathbb{P}(1 \text{ string missing}) - \mathbb{P}(2 \text{ strings missing}) \\
&= 1 - \mathbb{P}(\text{only 2 strings appear}) - \mathbb{P}(\text{only 1 string appears}) \\
&= 1 - 3 \cdot \left(\frac{2}{3}\right)^t - 3 \cdot \left(\frac{1}{3}\right)^t
\end{aligned}$$

We want this probability to be greater than or equal to 90%. The $\left(\frac{1}{3}\right)^t$ term goes to 0 faster than the $\left(\frac{2}{3}\right)^t$ term, so let's just ignore the $\left(\frac{1}{3}\right)^t$ term.

$$90\% = 0.9 \leq 1 - 3 \cdot \left(\frac{2}{3}\right)^t \implies 0.1 \leq 3 \cdot \left(\frac{2}{3}\right)^t \implies \frac{0.1}{3} \leq \left(\frac{2}{3}\right)^t$$
$$\implies \log_{\frac{2}{3}}(\frac{0.1}{3}) \leq t \implies 8.388 \leq t \implies t \approx 9$$

To verify, plug in $t = 9$: $1 - 3 \cdot \left(\frac{2}{3}\right)^9 - 3 \cdot \left(\frac{1}{3}\right)^9 \approx 0.922$. So basically, we Grover's 9 times. Then we should see all 3 satisfying strings with over 90% probability. This is all done in $O(9 \cdot n\sqrt{2^n}) \in O(\cdot n\sqrt{2^n})$ time.

68 Measuring/deleting one qubit of several (Lecture 18)

Video #068/100 Table of Contents

Say A,B,C are qubits, and you get them into state:

$$.9\,|000\rangle + .4\,|001\rangle - .1\,|010\rangle + .1\,|100\rangle + .1\,|111\rangle$$

Now say you do "extract C"

Law of 1-qubit Quantum Measurement: After measuring C,

- $\mathbb{P}$(display "0") = total squared amplitude on basic states where C = 0
- $\mathbb{P}$(display "1") = total squared amplitude on basic states where C = 1

And, consditioned on seeing the outcome C = 0, the new (unnormalized) state of A, B is the "part of" the old state where C = 0. Similarly, if seeing the outcome C = 1, the new (unnormalized) state of A, B is the part of the old state where C = 1.

- $\mathbb{P}$(display "0") = $.9^2 + (-.1)^2 + .1^2 = .83$.
  Then the unnormalized state that is left is $.9\,|00\rangle - .1\,|01\rangle + .1\,|10\rangle$
- $\mathbb{P}$(display "1") = $.4^2 + .1^2 = .17$.
  Then the unnormalized state that is left is $.4\,|00\rangle + .1\,|11\rangle$

68.1 Exercises

Three qubits A,B,C are in the joint state

$$-.1\,|000\rangle - .7\,|011\rangle + .1\,|100\rangle - .7\,|111\rangle$$

- We perform "extract A".
  What is the probability that the displayed outcome is "0"?
- Conditioned on the outcome of extract A being 0, now suppose we perform extract B.
  What is the probability that the displayed outcome is "0"?
- Further conditioned on the outcome of extract B being "0", now we perform extract C.
  What is the probability that the displayed outcome is "0"?

68.2 Solutions

- $\mathbb{P}(\text{"A = 0"}) = (-.1)^2 + (-.7)^2 = .50$

  Remaining unnormalized state is $-.1\,|00\rangle - .7\,|11\rangle$.
- $\mathbb{P}(\text{"B = 0"}\,|\,\text{"A = 0"}) = \frac{(-.1)^2}{(-.1)^2+(-.7)^2} = \frac{.01}{.50} = .02$

  Remaining unnormalized state is $-.1\,|0\rangle$.
- $\mathbb{P}(\text{"C = 0"}\,|\,\text{"A = 0"} \wedge \text{"B = 0"}) = \frac{(-.1)^2}{(-.1)^2} = 1$

69 The order of measuring qubits doesn't matter

Video #069/100 Table of Contents

We ought to check that "extract all" doing it qubit by qubit is the same as extracting from the original n-qubit state, and also check that the order we do the extracts don't matter.

Updated **Law of 1-qubit Quantum Measurement:** After measuring C,

- $\mathbb{P}$(display "0") = total fraction (to deal with unnormalized states) squared amplitude on basic states where C = 0
- $\mathbb{P}$(display "1") = total fraction squared amplitude on basic states where C = 1

I don't want to typeset the example out - just remember for destructive measurement, order of measurement doesn't matter.

69.1 Exercises

Give a formal proof of the following fact: let 3 qubits A B C be in some original state. Suppose we first extract A, then extract B. This will lead to 2 readout bits (4 possibilities), each with some probability, as well as new states for C.

Show that we get the same readout probabilities and new states for C if we instead had started from the original state and first measured B, then measured A.

69.2 Solutions

Let A B C be in state

$$a\,|000\rangle + b\,|001\rangle + c\,|010\rangle + d\,|011\rangle + e\,|100\rangle + f\,|101\rangle + g\,|110\rangle + h\,|111\rangle$$

where $a^2 + b^2 + c^2 + d^2 + e^2 + f^2 + g^2 + h^2 = 1$.

- Say the readout is $|00\rangle$. Then $A = 0, B = 0$. $\mathbb{P}(\text{C} = 0) = \frac{a^2}{a^2+b^2}$ and $\mathbb{P}(\text{C} = 1) = \frac{b^2}{a^2+b^2}$ either way.

  Extract A = 0. Remaining unnormalized state: $a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle$

  Extract B = 0. Remaining unnormalized state: $a\,|0\rangle + b\,|1\rangle$.

  Alternatively,

  Extract B = 0. Remaining unnormalized state: $a\,|00\rangle + b\,|01\rangle + e\,|10\rangle + f\,|11\rangle$.

  Extract A = 0. Remaining unnormalized state: $a\,|0\rangle + b\,|1\rangle$
- Say the readout is $|01\rangle$. Then $A = 0, B = 1$. $\mathbb{P}(\text{C} = 0) = \frac{c^2}{c^2+d^2}$ and $\mathbb{P}(\text{C} = 1) = \frac{d^2}{c^2+d^2}$ either way.

  Extract A = 0. Remaining unnormalized state: $a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle$

  Extract B = 1. Remaining unnormalized state: $c\,|0\rangle + d\,|1\rangle$.

  Alternatively,

  Extract B = 1. Remaining unnormalized state: $c\,|00\rangle + d\,|01\rangle + g\,|10\rangle + h\,|11\rangle$.

  Extract A = 0. Remaining unnormalized state: $c\,|0\rangle + d\,|1\rangle$

- Say the readout is $|10\rangle$. Then $A = 1, B = 0$. $\mathbb{P}(\mathrm{C} = 0) = \frac{e^2}{e^2+f^2}$ and $\mathbb{P}(\mathrm{C} = 1) = \frac{f^2}{e^2+f^2}$ either way.

  Extract A $= 1$. Remaining unnormalized state: $e\,|00\rangle + f\,|01\rangle + g\,|10\rangle + h\,|11\rangle$

  Extract B $= 0$. Remaining unnormalized state: $e\,|0\rangle + f\,|1\rangle$.

  Alternatively,

  Extract B $= 0$. Remaining unnormalized state: $a\,|00\rangle + b\,|01\rangle + e\,|10\rangle + f\,|11\rangle$.

  Extract A $= 1$. Remaining unnormalized state: $e\,|0\rangle + f\,|1\rangle$

- Say the readout is $|11\rangle$. Then $A = 1, B = 1$. $\mathbb{P}(\mathrm{C} = 0) = \frac{g^2}{g^2+h^2}$ and $\mathbb{P}(\mathrm{C} = 1) = \frac{h^2}{g^2+h^2}$ either way.

  Extract A $= 1$. Remaining unnormalized state: $e\,|00\rangle + f\,|01\rangle + g\,|10\rangle + h\,|11\rangle$

  Extract B $= 1$. Remaining unnormalized state: $g\,|0\rangle + h\,|1\rangle$.

  Alternatively,

  Extract B $= 1$. Remaining unnormalized state: $c\,|00\rangle + d\,|01\rangle + g\,|10\rangle + h\,|11\rangle$.

  Extract A $= 1$. Remaining unnormalized state: $g\,|0\rangle + h\,|1\rangle$

70 Tensor Product

Video #070/100 Table of Contents

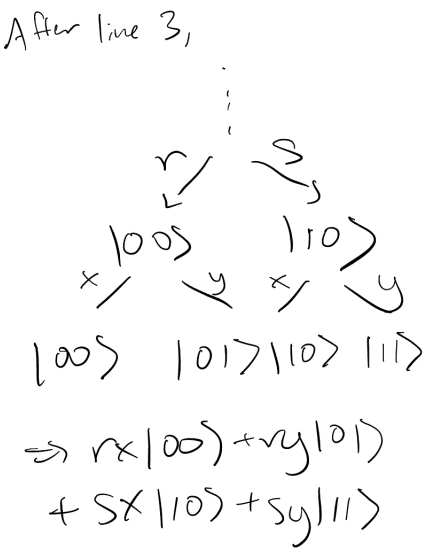
Suppose Alice creates a qubit A in her lab and gets into $r|0\rangle + s|1\rangle = \begin{bmatrix} r \\ s \end{bmatrix}$ and Bob creates B in his lab with state $x|0\rangle + y|1\rangle = \begin{bmatrix} x \\ y \end{bmatrix}$.

We're going to do some joint 2-qubit instruction on qubits A B. What is the joint state of A B? We'll notate it as $(r|0\rangle + s|1\rangle) \otimes (x|0\rangle + y|1\rangle)$, where $\otimes$ is the "tensor product" (or simply, "tensor") operator.

Say Alice did "new qubit A" and then some 1-qubit unitary U with $U|0\rangle = r|0\rangle + s|1\rangle$ and Bob did some unitary V with $V|0\rangle = x|0\rangle + y|1\rangle$.

1. new qubit A
2. do U on A
3. new qubit B
4. do V on B

After line 3,



Algebraic Facts for $\otimes$

- behaves like "non-commutative multiplication", i.e. $|u\rangle \otimes |v\rangle \neq |v\rangle \otimes |u\rangle$
- is commutative vis-a-vis scalars, i.e. $|u\rangle \otimes c|v\rangle = c|u\rangle \otimes |v\rangle = c(|u\rangle \otimes |v\rangle)$
- distributive across addition, i.e. $|u\rangle \otimes (|v\rangle + |w\rangle) = |u\rangle \otimes |v\rangle + |u\rangle \otimes |w\rangle$
- $(r|0\rangle) \otimes (x|0\rangle) + (r|0\rangle) \otimes (y|1\rangle) + (s|1\rangle) \otimes (x|0\rangle) + (s|1\rangle) \otimes (y|0\rangle)$
  $= rx(|0\rangle \otimes |0\rangle) + ry(|0\rangle \otimes |1\rangle) + sx(|1\rangle \otimes |0\rangle) + sy(|1\rangle \otimes |1\rangle)$
  $= rx|00\rangle + ry|01\rangle + sx|10\rangle + sy|11\rangle$

Suppose A, B are in joint state $|f\rangle \otimes |g\rangle$, then we do a 1-qubit unitary W on A. What's the new state?

1. Could do amplitude trees, or

2. Just do W on $|f\rangle$ first then do $W|f\rangle \otimes |g\rangle$. Think of it as Alice could've applied W to A before bringing A and B together, or could apply W after bringing A and B together.

What is the dot product of $|f\rangle \otimes |g\rangle$ and $|p\rangle \otimes |q\rangle$? It's actually $\langle f|p\rangle \cdot \langle g|q\rangle$ (the dot is regular multiplication of scalars).

70.1 Exercises

I am not gonna typeset this one. Ain't no way.

In class we introduced the tensor product of two column vectors; e.g.,

$$\text{for } |u\rangle = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad |v\rangle = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}, \quad \text{we have } |u\rangle \otimes |v\rangle = \begin{bmatrix} x_0 \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} \\ x_1 \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} \\ x_2 \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} \\ x_3 \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} x_0y_0 \\ x_0y_1 \\ x_0y_2 \\ x_1y_0 \\ x_1y_1 \\ x_1y_2 \\ x_2y_0 \\ x_2y_1 \\ x_2y_2 \\ x_3y_0 \\ x_3y_1 \\ x_3y_2 \end{bmatrix}.$$

(In quantum computing, we usually only encounter vectors whose dimension is a power of 2, but you can define tensor product for vectors of any dimension.)

In fact, there is a more general definition of the tensor product of any two matrices, which you can infer from the following example:

$$\text{for } U = \begin{bmatrix} x_{00} & x_{01} & x_{02} \\ x_{10} & x_{11} & x_{12} \end{bmatrix}, \quad V = \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix}$$

we have ... (wait a sec...)

$$U \otimes V = \begin{bmatrix} x_{00}V & x_{01}V & x_{02}V \\ x_{10}V & x_{11}V & x_{12}V \end{bmatrix} = \begin{bmatrix} x_{00} \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix} & x_{01} \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix} & x_{02} \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix} \\ x_{10} \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix} & x_{11} \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix} & x_{12} \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} x_{00}y_{00} & x_{00}y_{01} & x_{01}y_{00} & x_{01}y_{01} & x_{02}y_{00} & x_{02}y_{01} \\ x_{00}y_{10} & x_{00}y_{11} & x_{01}y_{10} & x_{01}y_{11} & x_{02}y_{10} & x_{02}y_{11} \\ x_{10}y_{00} & x_{10}y_{01} & x_{11}y_{00} & x_{11}y_{01} & x_{12}y_{00} & x_{12}y_{01} \\ x_{10}y_{10} & x_{10}y_{11} & x_{11}y_{10} & x_{11}y_{11} & x_{12}y_{10} & x_{12}y_{11}. \end{bmatrix}$$

In this definition, U can be any $p \times q$ matrix and V can be any $m \times n$ matrix; then $U \otimes V$ is a $pm \times qn$ matrix.

In class we said that if $|u\rangle$ is the state of some qubits, and $|v\rangle$ is the state of some other qubits, and you bring all these qubits together, the joint state is $|u\rangle \otimes |v\rangle$. So that explains why one would introduce the tensor product of two vectors.

But why would you care about the tensor product of two (square) matrices? Well, if U is a unitary operation performed on one block of qubits, and V is a unitary operation performed on a completely separate block of qubits (*not* necessarily unentangled from the first block), the giant unitary corresponding to doing *both* these operations is $U \otimes V$.

There are a few facts you need to know about tensor products:

(i) $U \otimes (V \otimes W) = (U \otimes V) \otimes W$.

(ii) $(U \otimes V)^\dagger = U^\dagger \otimes V^\dagger$. (In other words, the "reverse" of doing

"U on block one and also V on block two"

is to do the "reverse of U" on block one and also the "reverse of V" on block two.)

(iii) $(U_1 + U_2) \otimes V = U_1 \otimes V + U_2 \otimes V$, and similarly $U \otimes (V_1 + V_2) = U \otimes V_1 + U \otimes V_2$.

(iv) $(U \otimes V) \cdot (X \otimes Y) = (U \cdot X) \otimes (V \cdot Y)$. In particular, if $X = |u\rangle$ and $Y = |v\rangle$, this tells you that $(U \otimes V)(|u\rangle \otimes |v\rangle) = (U|u\rangle) \otimes (V|v\rangle)$. This can be read as saying: "if $|u\rangle$ and $|v\rangle$ were unentangled, and then you separately apply U to $|u\rangle$ and V to $|v\rangle$, then they remained unentangled, in the states $U|u\rangle$ and $V|v\rangle$".

Regarding the above four facts, it is not too hard to convince yourself of the first three. You can also just grind out the fourth, most important one, fact if you want. But in this problem, we'll once again illustrate it with "path diagrams".

Recall a much earlier exercise... We have an amber-colored bipartite graph, with m vertices on top, n vertices on bottom, and some directed edges going from top to bottom. Let A be the $n \times m$ matrix with nonnegative integer entries in which the entry in the ith column and jth row is the number of edges from vertex i (on top) to vertex j (on bottom). Similarly, we have a blue-colored graph with n vertices on top, p vertices on bottom, and an associated $p \times n$ matrix. As we know, in the $p \times m$ matrix $C = B \cdot A$, the entry in the ith column and kth row is the number of ways a walker can get from i to k when you glue the amber and blue graphs together along the n vertices.

(a) Temporarily forget about the blue graph. Instead, supposed we have another scarlet-colored bipartite graph with m' vertices on top, n' vertices on bottom, and associated $n' \times m'$ matrix S. Suppose Tweedledee stands on a top vertex of the amber graph, and Tweedledum stands on a top vertex of the scarlet graph. Now imagine that they can take a "joint step", which consists of Tweedledee following some amber edge and Tweedledum following some scarlet edge. Write the dimensions of the matrix $A \otimes S$, and explain how it encodes the number of possibilities for the twins' joint step.

(b) Let's bring the blue graph back now, and moreover let's introduce a teal-colored graph, with n' vertices on top, p' vertices on bottom, and associated $p' \times n'$ matrix T. Note that this teal graph can naturally be glued to the scarlet graph. Explain what $(B \otimes T) \cdot (A \otimes S)$ means, what $(B \cdot A) \otimes (T \cdot S)$ means, and why they are the same.

70.2 Solutions

(a) $A \otimes S$ is going to have $n \cdot n'$ rows and $m \cdot m'$ columns. It basically encodes every possible combination of joint steps for the twins as follows. A encodes Tweedledee. For every possible move Tweedledee can take, put the matrix that encodes Tweedledum's possible moves. Refer to the definition of tensor product for matrices (not vectors).

(b) $(B \otimes T) \cdot (A \otimes S)$ means the number of ways Tweedledee and Tweedledum can joint step past the amber/scarlet level, times the number of ways Tweedledee and Tweedledum can joint step past the blue/teal level.

In contrast, $(B \cdot A) \otimes (T \cdot S)$ means the number of ways Tweedledee can get from amber to blue in joint step with Tweedledum who's traveling from scarlet to teal. Both are the same; we're just trying to find the number of ways Tweedledee and Tweedledum can complete their respective goals of going from amber to blue and scarlet to teal in joint step with one another.

71 Entanglement

Video #071/100 Table of Contents

Entanglement: if A B are qubits in a state of the form $|f\rangle \otimes |g\rangle$, we say they're un-entangled. Otherwise, they are entangled. For example,

- $|10\rangle = |1\rangle \otimes |0\rangle$, so this is an unentangled state. There's no joint instructions. It's like new qubit A, new qubit B, toggle A.
- The uniform superposition state is also unentangled; it's just $|+\rangle \otimes |+\rangle$. It's like new qubit A, new qubit B, Hadamard A, Hadamard B. There's no joint instructions.
- Einstein-Podolski-Rosen state (or unofficially, **E**ntangled **P**ai**R**): $|EPR\rangle = \sqrt{\frac{1}{2}}|00\rangle + \sqrt{\frac{1}{2}}|11\rangle$. AFSOC it's unentangled. Then there are r, s, x, y such that $(r, s) \otimes (x, y) = (\sqrt{\frac{1}{2}}, 0, 0, \sqrt{\frac{1}{2}}) = (rx, ry, sx, sy)$. But then that would mean $ry(sx) = 0$ and $rx(sy) = \sqrt{\frac{1}{2}}^2 = \frac{1}{2}$, which would mean $rsxy = 0$ and $\frac{1}{2}$ simultaneously. Contradiction reached.

  Q: What is the state vector for just A by itself? Trick question! There is none.

Defn: more generally, one block of qubits $A_1, A_2, \ldots, A_n$ is un-entangled with another block $B_1, B_2, \ldots B_n$ if and only if the joint state is $|F\rangle \otimes |G\rangle$ where $|F\rangle$ is a 2^m dimensional state vector and $|G\rangle$ is a 2^n dimensional state vector. Basically, two sets of qubits are unentangled if we can make the A's and B's separately.

Example: $m = 2, n = 1$. Say we have qubits A_1, A_2, B in joint state $\sqrt{\frac{1}{2}}|001\rangle + \sqrt{\frac{1}{2}}|111\rangle$. Since $|001\rangle = |00\rangle \otimes |1\rangle$ and $|111\rangle = |11\rangle \otimes |1\rangle$, we can factor the joint state as $\left(\sqrt{\frac{1}{2}}|00\rangle + \sqrt{\frac{1}{2}}|11\rangle\right) \otimes |1\rangle$, which is just $|EPR\rangle \otimes |1\rangle$. So Alice's qubits are entangled with each other, but they are unentangled with Bob's qubits.

71.1 Exercises (continued from last one)

(c) Suppose we have a yellow-colored graph with m'' vertices on top, n'' vertices on bottom, and associated $n'' \times m''$ matrix Y. What story would you tell to explain why $(A \otimes S) \otimes Y = A \otimes (S \otimes Y)$ (and what are the dimensions of these matrices?)

(d) What story would you tell to explain why $(A \otimes S)^\dagger = A^\dagger \otimes S^\dagger$ (and what are the dimensions of these matrices?)

71.2 Solutions

(c) The dimensions are $n \cdot n' \cdot n''$ rows and $m \cdot m' \cdot m''$ columns. Story: $(A \otimes S) \otimes Y$ is Tweedledee and Tweedledum going in joint step together, and then there's some outside guy Tweedlejoe who's moving through the yellow colored graph in joint step with the duo Tweedledee and Tweedledum. $A \otimes (S \otimes Y)$ is where Tweedledee is moving through the amber graph by himself,

(d) The dimensions are $m \cdot m'$ rows and $n \cdot n'$ columns. From Video #017/100, exercise (c), we know the conjugate transpose of a paths matrix is like reversing the arrows in the corresponding graphs. So the story we'd tell is imagine we want Tweedledee and Tweedledum to reverse their paths and get back to where they started. Then you can either reverse their entire journey, or reverse each of their individual journeys and then have them do joint steps through the reversed journeys.

72 Ops on separate qubits: order doesn't matter (Lecture 19)

Video #072/100 Table of Contents

Wrapping up last lecture: Say A B are in the state $|s\rangle = w|00\rangle + x|01\rangle + y|10\rangle + z|11\rangle$. Say we do "extract B".

Then $\mathbb{P}(\text{B} = 0) = w^2 = y^2$ and $\mathbb{P}(\text{B} = 1) = x^2 + z^2$. It's nice to compute this by factoring $|s\rangle$ into $(|w\rangle|0\rangle + y|1\rangle) \otimes |0\rangle + (x|0\rangle + z|1\rangle) \otimes |1\rangle$.

Call $|\ell\rangle = (|w\rangle|0\rangle + y|1\rangle) \otimes |0\rangle$ and call $|r\rangle = (x|0\rangle + z|1\rangle) \otimes |1\rangle$. Note $|\ell\rangle$ and $|r\rangle$ are not unit vectors. But notice $\mathbb{P}(\text{B} = 0) = \langle\ell|\ell\rangle = w^2 + y^2$ and the leftover state is $|\ell\rangle$, and $\mathbb{P}(\text{B} = 1) = \langle r|r\rangle = x^2 + z^2$ and the leftover state is $|r\rangle$.

Then starting with $|s\rangle$, if we do

1. extract B
2. do unitary U on A

Split into cases: the result is

- With probability length$(\ell)^2$, outcome is "0" after line 1. Then the final unnormalized state of A is $U|\ell\rangle$.
- With probability length$(r)^2$, outcome is "1" after line 1. Then the final unnormalized state of A is $U|r\rangle$.

But what if we did the operations in the opposite order? Then starting with $|s\rangle$, if we do

1. do unitary U on A
2. extract B

After line 1, state is $(U|\ell\rangle) \otimes |0\rangle + (U|r\rangle) \otimes |1\rangle$.

Then after line 2, the probability of "0" is length$(U|\ell\rangle)^2$ =length$(|\ell\rangle)^2$ since unitary operations preserve length. Same follows for the case where we see "1".

Conclusion: operations done on separate sets of qubits can be equivalently be reordered. Think of this for physical qubits: Alice can take her qubit and do whatever op, and Bob can take his qubit away and do whatever op. Due to special relativity, for physically far apart objects, it doesn't even make sense to ask what event happens first. So if Alice and Bob are really far apart, it really doesn't matter which operation is done first.

72.1 Exercises: Dot product of tensor products

Let $|u\rangle$ and $|x\rangle$ have the same dimension, and let $|v\rangle$ and $|y\rangle$ have the same dimension. Prove that the dot product of $|u\rangle \otimes |v\rangle$ and $|x\rangle \otimes |y\rangle$ is equal to $\langle u|x\rangle \cdot \langle v|y\rangle$ where "$\cdot$" just denotes multiplication of two numbers.

You *must* do this problem exclusively using bra-ket notation. No writing out coordinates for the vectors. You can and should use (without proof) properties (i)-(iv) from the last exercises. Explain carefully how/when you use each property. You can and also should use that 1×1 matrices can be freely type-casted to scalars (and point out when you use this).

72.2 Solutions

$$\begin{aligned}
\big\langle |u\rangle \otimes |v\rangle \big| |x\rangle \otimes |y\rangle \big\rangle &= (|u\rangle \otimes |v\rangle)^\dagger \cdot (|x\rangle \otimes |y\rangle) && \text{(defn of inner/dot product)} \\
&= (|u\rangle^\dagger \otimes |v\rangle^\dagger) \cdot (|x\rangle \otimes |y\rangle) && \text{(ii)} \\
&= (|u\rangle^\dagger |x\rangle) \otimes (|v\rangle^\dagger |y\rangle) && \text{(iv)} \\
&= (\langle u|x\rangle) \otimes (\langle v|y\rangle) && \text{(defn of inner/dot product)} \\
&= (\langle u|x\rangle) \cdot (\langle v|y\rangle) && (\otimes \text{ and } \cdot \text{ are the same for } 1 \times 1 \text{ matrices}) \\
&= \langle u|x\rangle \cdot \langle v|y\rangle && \text{(typecast } \langle u|x\rangle \text{ and } \langle v|y\rangle \text{ to scalars)}
\end{aligned}$$

73 Quantum teleportation: part 1

Video #073/100 Table of Contents

Most famous entangled state for 2 qubits: $|EPR\rangle = |00\rangle + |11\rangle$ (unnormalized). To make this state:

1. new qubit A $1|0\rangle + 0|1\rangle$
2. Add&Diff A $1|0\rangle + 1|1\rangle$
3. new qubit B $1|00\rangle + 1|10\rangle$
4. if A then toggle B $1|00\rangle + 1|11\rangle$

   line 4 is a joint operation
5. Alice and Bob "depart" (maybe Alice goes to CMU and Bob goes to Pitt!)

Quantum teleportation means we can teleport information between qubits that are really far apart. Imagine Bob went back to his lab. His boss Charlie comes in, and hands Bob a new qubit C. Charlie wants Bob to transfer the state of this qubit C to Alice!

Say Charlie made C in state $x|0\rangle + y|1\rangle$. Bob *could* just walk from Pitt to CMU and give Alice the qubit, or

- if Bob knew x and y, he could just call Alice and just make a bunch of qubits in that state. Unfortunately, x and y are arbitrary real numbers such that $x^2 + y^2 = 1$, so it would be hard to send this information over with perfect precision (practically, could just send it to a desired precision).
- Unfortunately, Bob doesn't even know x and y. But recall that Alice and Bob have this entangled state between their qubits A and B.

Teleportation Sketch:

0. Pre-share an $|EPR\rangle$
1. Bob does some qubit manipulations on B and C.
2. Because of no-learning principle, we know that qubits B and C have to get "ruined" somehow, otherwise you could just keep helping Alice make infinite copies of C. So Bob does "extract B C" getting classical bits b c

   A is almost in the right state $x|0\rangle + y|1\rangle$
3. Bob texts Alice the classical bits b c
4. Alice uses b c to do some operations to A and fix it up to the desired state $x|0\rangle + y|1\rangle$

Notes:

- The only qubit left afterwards is A. The $|EPR\rangle$ pair is also ruined.
- Bob only texts 2 bits, way fewer than Bob would send even if he knew x and y precisely. This is the only stuff that physically moves from Bob's lab to Alice's lab
- No physical qubit gets "teleported", just states change
- Neither Bob nor Alice ever learns x or y.

Funnily enough, when they did this entanglement, they could only do the experiment once a month because if the moon wasn't a new moon, the moonlight would mess up the entanglement when shooting the photon from Earth up to a satellite.

73.1 Exercises

Alice creates a qubit A in her own lab, in the state $.6\,|0\rangle + .8\,|1\rangle$. Bob creates a qubit B in his own lab, in the state $.6\,|0\rangle - .8\,|1\rangle$. What is the joint state of A B?

Suppose now that Bob does "extract B" and the displayed outcome is "1". What is the new (normalized) state of A?

True or false?
The 2-qubit state $.8\,|10\rangle + .6\,|11\rangle$ is entangled.
The 2-qubit state $|+\rangle \otimes |-\rangle$ is entangled.

73.2 Solutions

Joint (unnormalized) state of A B is $(.6\,|0\rangle + .8\,|1\rangle) \otimes (.6\,|0\rangle - .8\,|1\rangle) = .36\,|00\rangle - .48\,|01\rangle + .48\,|10\rangle - .64\,|11\rangle$

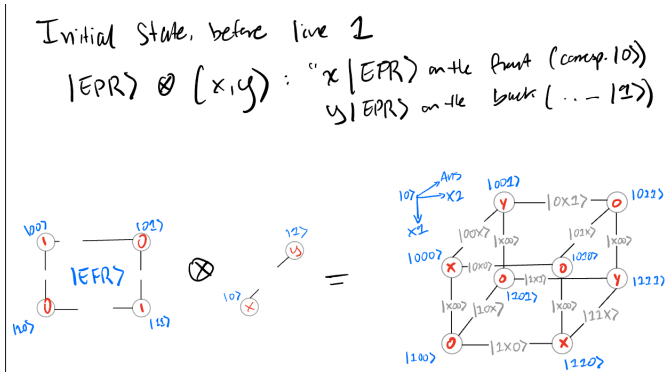
Bob does extract B and sees 1. The remaining state for A is the amplitudes on the states where B = 1, so we're left with $-.48\,|0\rangle - .64\,|1\rangle$. We need to normalize this; the ratio $-.48 : -.64$ is equal to the ratio $.6 : .8$. We maintain the negative amplitudes to find that the remaining state for A is $-.6\,|0\rangle - .8\,|1\rangle$

False. Consider $(0\,|0\rangle + 1\,|1\rangle) \otimes (.8\,|0\rangle + .6\,|1\rangle) = 0\,|00\rangle + 0\,|01\rangle + .8\,|10\rangle + .6\,|11\rangle = .8\,|10\rangle + .6\,|11\rangle$. We can get to both component states using unitary operations, so this 2-qubit state is the tensor product of two states, so its unentangled.
False; by definition it's just the tensor product of two states, so its unentangled.

74 Quantum teleportation: part 2

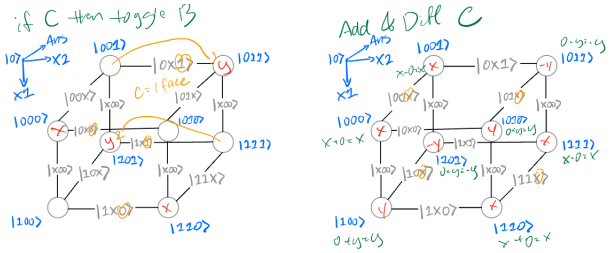
Video #074/100 Table of Contents

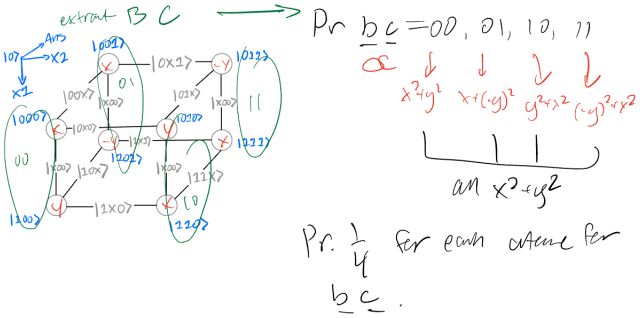
Initial state before line 1 is
$|EPR\rangle \otimes (x\,|0\rangle + y\,|1\rangle) = (1\,|00\rangle + 1\,|11\rangle) \otimes (x\,|0\rangle + y\,|1\rangle) = x\,|000\rangle + y\,|001\rangle + x\,|110\rangle + y\,|111\rangle$



Bob's actions are

1. if C then toggle B
2. Add&Diff C
3. b c = extract B C
4. transmit b c to Alice





Alice's Actions to fix:

- bc = 00, corresponds to $x\,|0\rangle + y\,|1\rangle$ - do nothing
- bc = 01, corresponds to $x\,|0\rangle - y\,|1\rangle$ - if A then minus
- bc = 10, corresponds to $y\,|0\rangle + x\,|1\rangle$ - toggle A
- bc = 11, corresponds to $-y\,|0\rangle + x\,|1\rangle$ - toggle A, then if A then minus

74.1 Exercises: Teleporting entanglement?

Consider the Quantum Teleportation scenario described in class, but with a twist. Unbeknownst to Bob, the qubit C that Charlie gives to him is actually (potentially) entangled with some other qubit D. That is, Charlie had previous created two qubits C, D in a joint state $p\,|00\rangle + q\,|01\rangle + r\,|10\rangle + s\,|11\rangle$, put D in their pocket, then handed C off to Bob with the usual "Transfer the state of this qubit to Alice!" command. Bob and Alice run the Quantum Teleportation protocol as normal, at the end of which there are two qubits left: A and D. True or false: the joint state of A, D is $p\,|00\rangle + q\,|01\rangle + r\,|10\rangle + s\,|11\rangle$. If true, give a proof. If false, work thorugh a counterexample.

74.2 Solutions

False. In the beginning, we know AB is in state $|EPR\rangle$ and CD is in state $p\,|00\rangle + q\,|01\rangle + r\,|10\rangle + s\,|11\rangle$. So ABCD are in one huge joint state,

$|EPR\rangle \otimes (p\,|00\rangle + q\,|01\rangle + r\,|10\rangle + s\,|11\rangle)$

$(1\,|00\rangle + 1\,|11\rangle) \otimes (p\,|00\rangle + q\,|01\rangle + r\,|10\rangle + s\,|11\rangle) = (1,0,0,1) \otimes (p,q,r,s) = (p,q,r,s,0,0,0,0,0,0,0,0,p,q,r,s)$

So the state is:
$p\,|0000\rangle + q\,|0001\rangle + r\,|0010\rangle + s\,|0011\rangle + p\,|1000\rangle + q\,|1001\rangle + r\,|1010\rangle + s\,|1011\rangle$

Now Bob does if C then toggle B. So the state is
$p\,|0000\rangle + q\,|0001\rangle + r\,|0110\rangle + s\,|0111\rangle + p\,|1000\rangle + q\,|1001\rangle + r\,|1110\rangle + s\,|1111\rangle$

Now Bob does Add&Diff C. So the state is
$(p+0)\,|0000\rangle + (p-0)\,|0010\rangle + (q+0)\,|0001\rangle + (q-0)\,|0011\rangle + (0+r)\,|0100\rangle + (0-r)\,|0110\rangle + (0+s)\,|0101\rangle +$
$(0-s)\,|0111\rangle + (p+0)\,|1000\rangle + (p-0)\,|1010\rangle + (q+0)\,|1001\rangle + (q-0)\,|1011\rangle + (0+r)\,|1100\rangle + (0-r)\,|1110\rangle +$
$(0+s)\,|1111\rangle + (0-s)\,|1111\rangle$
$= (p,q,p,q,r,s,-r,-s,p,q,p,q,r,-s,-r,s)$

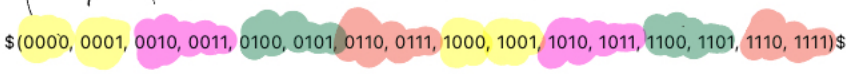
Now Bob does extract B C.
For my reference, the binary numbers from 0-15 are
$(0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111)$

- bc = 00 leaves state (p,q,p,q)
- bc = 01 leaves state (p,q,p,q)
- bc = 10 leaves state (p,q,p,q)
- bc = 11 leaves state (p,q,p,q)

The diagram on the next page summarizes this:

p q r s 0 0 0 0 0 0 0 0 p q r s

⇒ p q pq rs -r-s pq pq r-s-r s



bc	final state (00, 01, 10, 11)
00	(p, q, p, q)
01	(p, q, p, q)
10	(r, s, r, -s)
11	(r, s, -r, s)

→ not equal to (p, q, r, s).

So false.

75 The key property of $|EPR\rangle$

Video #075/100 Table of Contents

Starting from magical $|00\rangle + |11\rangle$,

Say Alice does toggle A. Then the new state is $|10\rangle + |01\rangle = |01\rangle + |10\rangle$.

On the other hand, say Bob does toggle B. Then the new state is $|01\rangle + |10\rangle$. These are the same! Alice can remotely do "toggle B" by merely doing "toggle A".

Cool fact: starting from $|EPR\rangle$, if Alice does U on A, it's equivalent to Bob doing $U^\dagger$ on B.

- transpose of U for real matrices
- reversed path diagram
- becomes $U^\dagger$ if U is complex. I'll use the more general $U^\dagger$.
- same as "undo-U" or inverse of U.

Notation: if you have a 2×2 matrix $\begin{bmatrix} w & x \\ y & z \end{bmatrix}$, we write $\text{vec}(M) = (w, x, y, z)$

Suppose two qubits A B are in state $\text{vec}(M)$.

- Fact 1: Suppose Alice does U on Am where U is 1-qubit unitary. Then the new state is $\text{vec}(UM)$ where UM is an easy matmul of two 2×2 matrices. A simple proof is $\begin{bmatrix} U \end{bmatrix} \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} U \begin{bmatrix} w \\ y \end{bmatrix} & U \begin{bmatrix} x \\ z \end{bmatrix} \end{bmatrix}$
- Fact 2: Suppose Bob does V on B, where V is 1-qubit unitary. Then the new state is $\text{vec}(MV^T)$. Hinges on the idea that $(VM^T)^T = MV^T$
- both can be proved visually using a little square that represents a 2-qubit state, then applying operations on it. The video proof is kinda confusing for me; I prefer the linalg explanation.

Recall: $|EPR\rangle = |00\rangle + |11\rangle = \text{vec}(I)$.
By Fact 1, if A B starts in $|EPR\rangle$, and Alice does U on A, then the new state is $\text{vec}(UI) = \text{vec}(U)$.
By Fact 2, if A B starts in $|EPR\rangle$, and Bob does V on B, then the new state is $\text{vec}(IV^\dagger) = \text{vec}(V^\dagger)$. If V were the undo of U, then $V = U^\dagger \implies V^\dagger = U$. So Bob doing $U^\dagger$ is the same as Alice doing U.

75.1 Exercises

(a) The usual way we make the state $|EPR\rangle$ is to start with two qubits A B in the basic state $|00\rangle$ then do

`MakeEPR`

(a) Add&Diff A

(b) if A then toggle B

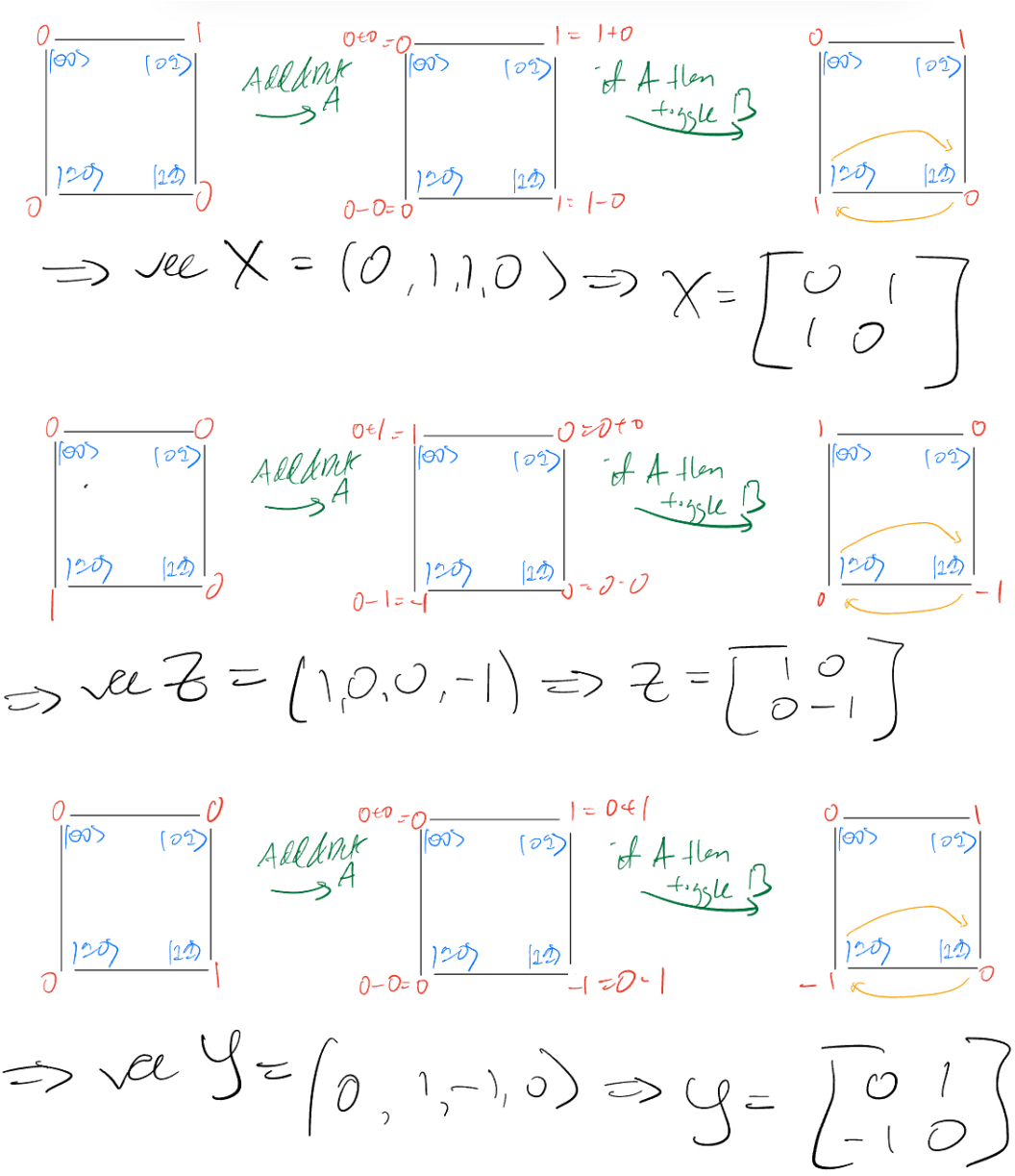
As you know, when we run MakeEPR (technically not unitary, since it makes unnormalized states; we can get a 4×4 unitary by doing $\sqrt{\frac{1}{2}}$MakeEPR), we get $|EPR\rangle = \text{vec}(I)$ where I is the 2×2 identity matrix. Now we would also run MakeEPR on $|01\rangle$, $|10\rangle$, or $|11\rangle$. Let's denote the results of these by $\text{vec}(X)$, $\text{vec}(Z)$, and $\text{vec}(Y)$ respectively (yes, X, Z, Y is an intended ordering). Work out what X, Y, Z are by tracing through MakeEPR on each of the three starting states.

Hint: by virtue of MakeEPR using Add&Diff, not Hadamard, all the entries will be 0 or ± 1.

(b) People who love Quantum Mechanics love the four matrices I, X, Y, Z (actually, they prefer to use iY in place of Y since that makes $Y^\dagger = Y$ so all the matrices are Hermitian). One reason is that they happen to be unitary matrices! As such, you can interpret them as rotation/reflection operations on $\mathbb{R}^2$. Describe in words how I, X, Y, Z operate.

75.2 Solutions

(a) See image



(b) I is a no-op, X is a reflection across the line $\theta = 45°$ (line $y = x$), Y is rotation by 90° clockwise, and Z is a reflection across the line $\theta = 0°$ (line $y = 0$, or just the x-axis).

76 How Alice and Bob mutually rotate $|EPR\rangle$ (Lecture 20)

Video #076/100 Table of Contents

Recall $|EPR\rangle = |00\rangle + |11\rangle$

The nature of reality. EPR stands for Einstein-Podolski-Rosen. They thought there was a problem in quantum mechanics.

Any experiment done in a physical location L should be determined by some real physical thing in the vicinity of L (local realism). However, our laws of quantum mechanics seem to violate this - entanglement shows that nature weirdly remembers even when things are taken apart.

Quantum Code

1. Alice & Bob jointly make A B in state $|EPR\rangle$.
2. Alice takes A, Bob takes B, they depart
3. Alice measures A
4. Bob measure B

Once Alice measures A, Bob's outcome is predetermined to be the same as whatever Alice saw. This isn't too weird; consider this Classical Code

1. a = randomBit(); b = a;
2. Alice takes a (without looking at it), Bob takes b (without looking at it), they depart
3. Alice looks at a
4. Bob looks at b

In the classical code, the values of a are determined after line 1. But in the quantum code, A and B don't have a fixed value until step 3. Einstein-Podolski-Rosen thought that qubits behaved more like the classical code - somehow, before step 3, local realism dictates what A and B are, then Alice and Bob just measure this later. But modern quantum mechanics is different - the value of A and B is fixed after step 3, not before. Hence local realism can't exist (John Bell experiment, simplified into the CHSH Game).

- if qubits A B are in any state vec(M) and you do U on A, then the new state is vec(UM), or if you do V on B instead, the new state is vec($MV^\dagger$)
- Say the initial state of A B is $|EPR\rangle = \text{vec}(I)$.
- Alice does Rot_α on A for some angle α
- Bob does Rot_β on B for some angle β

  these are operations on separate qubits, so the order doesn't matter.
- The new state is $\text{vec}(Rot_\alpha \cdot I \cdot Rot_\beta^\dagger) = \text{vec}(Rot_\alpha Rot_{-\beta}) = \text{vec}(Rot_{\alpha-\beta})$
- Notation: $|EPR\rangle_\theta = \text{vec}(Rot_\theta) = \text{vec}\begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix} = (\cos\theta)|00\rangle + (-\sin\theta)|01\rangle + (\sin\theta)|10\rangle + (\cos\theta)|11\rangle$. In particular, $|EPR\rangle_0 = |EPR\rangle$.

Q: Say A, B are in state $|EPR_\theta\rangle$, and Alice does $Rot_{\alpha'}$ on A. New state?

A: $\text{vec}(Rot_{\alpha'} \cdot Rot_\theta) = \text{vec}(Rot_{\theta+\alpha'})$.

Similarly, if Bob instead did $Rot_{\beta'}$, we get $|EPR_{\theta-\beta'}\rangle$

So what happens when you extract all on $|EPR_\theta\rangle$?

- Note $|EPR_\theta\rangle$ is unnormalized for now; the total squared amplitude is $2(\sin^2 + \cos^2) = 2$

measurement chart

A \ B	0	1
0	$\frac{1}{2}\cos^2\theta$	$\frac{1}{2}\sin^2\theta$
1	$\frac{1}{2}\sin^2\theta$	$\frac{1}{2}\cos^2\theta$

  - $\mathbb{P}$(A outcome = B outcome) is $\cos^2\theta$
  - $\mathbb{P}$(A outcome $\neq$ B outcome) is $\sin^2\theta$
  - Random bits, but are correlated, not independent.
- Measuring $|EPR_0\rangle$ guarantees that Alice and Bob see the same outcome. The two bits are fully correlated.
- Alternatively, $|EPR_{90^\circ}\rangle$ guarantees that Alice and Bob are anticorrelated; Alice and Bob see different measurements.
- Measuring $|EPR_{45^\circ}\rangle$ makes the Alice and Bob bits independent; equal chance of being equivalent and not equivalent.

76.1 Exercises (continued from last one)

(c) Suppose Alice and Bob create A B in the $|EPR\rangle$ state; then Alice grabs A and goes to her office at CMU, and Bob grabs B and goes to his office at Pitt. Claim: Alice can now change the joint state to any of the 4-qubit state vec(I), vec(X), vec(Y), vec(Z) of her choice (by acting only on A of course.) Prove this claim.

(d) Suppose A B are photons, and there is a fiber optic cable between Alice and Bob's offices. Having changed the joint state to vec(P) for some $P \in \{I, X, Y, Z\}$, Alice physically sends A along the fiber-optic cable, and now Bob possesses both A and B. Prove that there is a way for Bob to figure out what P is with 100% probability.

Remark: Since P was any one of four possibilities (Alice's choice), this means that with the aid of an $|EPR\rangle$ pair, Alice can convey 2 classical bits of information to Bob by physically transmitting one qubit - sort of the exact opposite of Quantum Teleportation!

76.2 Solutions

(c) Current state $|EPR\rangle$ is already vec(I). So Alice can just do nothing.
Now note vec(I) $= 1\,|00\rangle + 0\,|01\rangle + 0\,|10\rangle + 1\,|11\rangle = |00\rangle + |11\rangle$.
To get to vec(X) $= 0\,|00\rangle + 1\,|01\rangle + 1\,|10\rangle + 0\,|11\rangle = |01\rangle + |10\rangle$, Alice can do toggle A.
To get to vec(Y) $= 0\,|00\rangle + 1\,|01\rangle - 1\,|10\rangle + 0\,|11\rangle = |01\rangle - |10\rangle$, Alice can Rotate A by 90 degrees clockwise.

$|0\rangle \to -|1\rangle$ and $|1\rangle \to |0\rangle$

To get to $\text{vec}(Z) = 1|00\rangle + 0|01\rangle + 0|10\rangle - 1|11\rangle = |00\rangle - |11\rangle$, Alice can do toggle A, then Rotate A by 90 degrees clockwise.

(d) Bob has A and B.

- Bob does if A then toggle B.

  $\text{I} \mapsto |00\rangle + |10\rangle$
  $\text{X} \mapsto |01\rangle + |11\rangle$
  $\text{Y} \mapsto |01\rangle - |11\rangle$
  $\text{Z} \mapsto |00\rangle - |10\rangle$

- Bob does Add&Diff A

  $\text{I} \mapsto 2|00\rangle$
  $\text{X} \mapsto 2|01\rangle$
  $\text{Y} \mapsto 2|11\rangle$
  $\text{Z} \mapsto 2|10\rangle$

- Basically, Bob has undone MakeEPR. He now extracts all.
  If he sees AB = 00, Alice picked I
  If he sees AB = 01, Alice picked X
  If he sees AB = 11, Alice picked Y
  If he sees AB = 10, Alice picked Z.

77 The CHSH Game

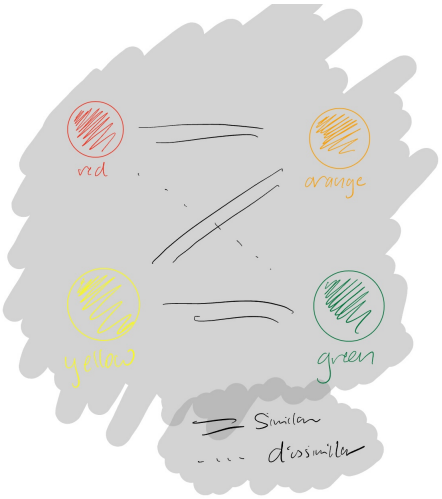
Video #077/100 Table of Contents

Cooperative game for Alice and Bob, with two referees.

CHSH: A co-op game for Alice and Bob

- Alice and Bob can plan ahead, but then they're separated by one kilometer.
- Alice's referee flips a coin whose sides are Red/Yellow. Alice must respond with $a \in \{0, 1\}$ within 1 nanosecond.
- Bob's referee flips a coin whose sides are Orange/Green. Bob must respond with $b \in \{0, 1\}$ within 1 nanosecond.
- Referees come together if Alice and Bob won

  Win condition: a and b have same similarity as the two colors they saw.



  Red and orange are similar, Orange and Yellow are similar, Yellow and Green are similar, but Red and Green are dissimilar.

  For example, $a = b$ and RO, YO, YG are wins. $a \neq b$ and RG is also a win.

The distance is just to ensure Alice can't text Bob what she sees before Bob has to make a decision.

77.1 Exercises

So what if you and your classmate were Alice and Bob? What strategy would you plan in advance to use? What would be your probability of winning?

77.2 Solutions

We should just agree to both pick the same bit. Then our probability of winning is 75%.

Proof. Possible coin flips are in the space $\Omega = \{R, Y\} \times \{O, G\} = \{RO, RG, YO, YG\}$. If we pick the same bit, say we both pick 1 no matter what, then we have a 75% chance of winning because RO, YO, YG are winning states because they are similar. □

78 Classical win probability of CHSH is 75%

Video #078/100 Table of Contents

What is the best strategy? Assuming their strategy is deterministic, the strategy they choose is just labeling each side of the coin with the bit each player would answer. There are 4 possible edges, 3 are wins if they are equal. So having both Alice and Bob always pick 0 is a good strategy. Another valid option is Alice says 0 on red, 1 on yellow and Bob always says 1. But either way, the best they can do is $\frac{3}{4}$ chance of winning.

Theorem: If Alice and Bob are classical and deterministic, their win probability is at most 75%. It's impossible to win $\frac{4}{4}$ times, but $\frac{1}{4}, \frac{2}{4}, \frac{3}{4}$ are possible.

Q: Could it help Alice and Bob to use randomness? e.g., preshare a bunch of coin flips.

A: Doesn't help though. Constructing all this randomness doesn't help because, even if they had like 20 random bits, well after they randomly decide the bits, they play deterministically. Then they can decide their probability of winning, which cannot exceed 75%

Theorem: By pre-sharing an $|EPR\rangle$ state, they have a winning strategy with probability a little over 85%. This disproves local realism - once Alice and Bob move apart, their qubits don't get determined at location L. Nature can sort of remember the entangled qubits. Weird stuff!

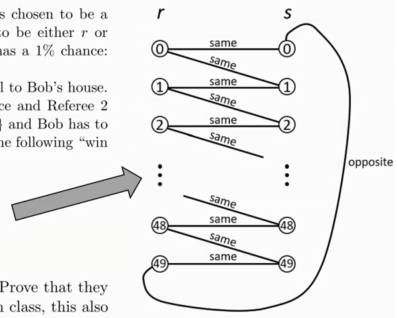
78.1 Exercises

Alice and Bob are going to play a cooperative game similar to the CHSH game.

It starts with Referee 1 and Referee 2 having a secret meeting in which r is chosen to be a random integer between 0 and 49 (inclusive), and then s is randomly chosen to be either r or $r+1$ mod 50. In other words, each of the following 100 possibilities for (r, s) has a 1% chance: $(0,0), (0,1), (1,1), (1,2), (2,2), \ldots, (48,49), (49,49), (49,0)$.

Now Alice and Referee 1 travel to Alice's house, and Bob and Referee 2 travel to Bob's house. These houses are 1km apart. At the stroke of noon, Referee 1 reveals r to Alice and Referee 2 reveals s to Bob. Within 1 nanosecond, Alice has to respond with a bit $a \in \{0,1\}$ and Bob has to respond with a bit $b \in \{0,1\}$. Thereafter, everyone gets together and they check the following "win condition":

Win condition: Alice and Bob win if $a = b$, *except* in the case $(r, s) = (49, 0)$, in which case they win if $a \neq b$.



Suppose that Alice and Bob play according to some deterministic strategy. Prove that they will lose the game with probability at least 1%. (Remark: as we discussed in class, this also means that even if they are allowed to generate and share random bits before the game starts, their probability of losing is still at least 1%.

78.2 Solutions

Proof. Alice and Bob will lose the game with probability at least 1%. There are 99 similar edges and one dissimilar one, namely $(r, s) = (49, 0)$. Say Alice and Bob do the dominant (deterministic) strategy from the CHSH game: picking 0 (or WLOG, 1) always. Then they'll always have $a = b$, so they win $\frac{99}{100} = 99\%$ of the time. There is no assignment that will get a probability of losing less than 1%. This is because, even if they use randomness, once the random bits have their values set, the game is determined. Alice and Bob play deterministically based on the randomness. And we saw from the previous bulletpoint that it's not possible to play deterministically with a loss percentage less than 1%. So when measuring the average loss probability across all the outcomes for the random bits, you won't end up with a loss percentage less than 1% (weighted average of a bunch of numbers that are at least 1% cannot be a number less than 1%. Hence the cumulative loss probability is at least 1%.) □

79 Quantum win probability of CHSH is 85%

Video #079/100 Table of Contents

What's the plan for 85% win using quantum? Alice is shown Red or Yellow, then rotate by a different angle depending on the outcome seen. Bob does the same, depending on whether he sees Orange or Green.

Strategy: start with entangled A B in $|EPR_0\rangle$.

- Alice sees

  Red, applies $Rot_{\alpha(R)}$ on A

  Yellow, applies $Rot_{\alpha(Y)}$ on A

- Bob sees

  Orange, applies $Rot_{\alpha(O)}$ on A

  Green, applies $Rot_{\alpha(G)}$ on A

Final joint state $|EPR_\theta\rangle$, There are four cases for the coins (measurement probabilities from Video #076/100)

RO Then $\theta = \alpha(R) - \alpha(O)$. Win iff a = b, has probability $\cos^2\theta$.
We want θ to be close to 0 so $\cos^2\theta$ is close to 1.

RG Then $\theta = \alpha(R) - \alpha(G)$. Win iff a $\neq$ b, has probability $\sin^2\theta$.
We want θ to be close to $\frac{\pi}{2}$ so $\sin^2\theta$ is close to 1.

YO Then $\theta = \alpha(Y) - \alpha(O)$. Win iff a = b, has probability $\cos^2\theta$.
We want θ to be close to 0 so $\cos^2\theta$ is close to 1.

YG Then $\theta = \alpha(Y) - \alpha(G)$. Win iff a = b, has probability $\cos^2\theta$.
We want θ to be close to 0 so $\cos^2\theta$ is close to 1.

So $\alpha(R), \alpha(O), \alpha(Y)$ should be pretty close together, and all should be roughly perpendicular to $\alpha(G)$. They can "spread out their wrongness" over the edges.

What if $\alpha(R) = 0°, \alpha(O) = 30°, \alpha(Y) = 60°, \alpha(G) = 90°$. Then the win probability is

RO $\theta = 0° - 30°$. $\cos^2(-30°) = \frac{3}{4}$

RG $\theta = 90° - 0° = 90°$. $\sin^2(90°) = 1$

YO $\theta = 60° - 30°$. $\cos^2(30°) = \frac{3}{4}$

YG $\theta = 60° - 90°$. $\cos^2(-30°) = \frac{3}{4}$

Weighted average of win probabilities is $\frac{13}{16} \approx 0.81$! Better than 75%, but,

More optimal strategy: $\alpha(R) = 0°, \alpha(O) = 22.5°, \alpha(Y) = 45°, \alpha(G) = 67.5°$

RO $\theta = 0° - 22.5°$. $\cos^2(-22.5°) \approx 0.85$

RG $\theta = 67.5° - 0° = 90°$. $\sin^2(67.5°) \approx 0.85$

YO $\theta = 45° - 22.5°$. $\cos^2(22.5°) \approx 0.85$

YG $\theta = 45° - 67.5°$. $\cos^2(-22.5°) \approx 0.85$

Weighted average of win probabilites is 0.85! Bell's inequality says that if you're classical, you can't win Bell's game (or the CHSH game) with win probability greater than 75%! Several Nobel Prizes (e.g., 1982, 2022) have been won for physically demonstrating this 85% win probability improvement using quantum.

79.1 Exercises (continued from last one)

On the other hand, suppose that before the game starts, Alice and Bob prepare qubits A, B in state $|EPR\rangle$, and then Alice keeps A and Bob keeps B. Describe and analyze a strategy for Alice and Bob where they lose with probability at most 0.1%. (There are slightly different ideas you could come up with, but you should definitely be able to achieve a strategy with losing probability at most $\left(\frac{\pi}{100}\right)^2$. Since $\pi^2 \leq 10$, this is indeed at most 0.1%. In fact, the best possible strategy loses with probability a little less than 0.025%).

79.2 Solutions

In the video, students initially thought of 4 angles by dividing 90° by $4 - 1 = 3$ (number of edges minus one). So we are splitting $\frac{\tau}{4}$ into 50 pieces, making each angle $\frac{\tau}{4 \cdot 50} = \frac{\tau}{200} \approx 1.8°$ apart. Keep in mind $\sin^2 88.2° \approx 0.999 \approx \cos^2 1.8°$. Let's try that and analyze it:

- Alice sees a number r. She rotates A by $r\frac{\tau}{200}$. Bob sees a number s. He rotates B by $s\frac{\tau}{200}$.
- Then the win probability if $(r, s) \neq (49, 0)$ is $\cos^2\left((r-s)\frac{\tau}{200}\right)$.

  There is 50% chance $r = s$, 50% chance $r = s{+}1 \bmod 50$. Since this is the case excluding $(r, s) = (49, 0)$, we know $r < s$, so $r - s = -1$ in 49 cases, and in 50 cases, $r = s$.

  So either the angle will be $(r-r)\frac{\tau}{200}$ or $\left(r-(r+1)\right)\frac{\tau}{200} = \frac{\tau}{200} = 1.8°$.

  $\mathbb{P}(win) = \cos^2 0° = 1$ or $\mathbb{P}(win) = \cos^2 1.8°$
- Then the win probability if $(r, s) = (49, 0)$ is $\sin^2\left((r-s)\frac{\tau}{200}\right)$.

  Alice will rotate by $49\frac{\tau}{200} = 88.2°$, Bob will not rotate at all (0°).

  $\mathbb{P}(win) = \sin^2 88.2°$
- Then the win probability is $\frac{1}{100}\left(50 \cdot 1 + 49\cos^2 1.8° + \sin^2 88.2°\right) \approx 0.9995$, meaning the probability of losing is about 0.05%, which is less than 0.1%.

80 Rotation Estimation: Bird's-Eye View (Lecture 21)

Video #080/100 Table of Contents

Recapping Lecture 17: Rotation Estimation

Given 1-qubit mystery rotation R, acting as rotate by $\theta \in \{0°, 90°\}$

Also given R^{10^k} for $k = 1, 2, 3, \ldots, n$

Goal: estimate θ to n digits of precision.

Sketch of algorithm:

0a Do C times: make $|0\rangle$, apply R, extract

0b Say t of the outcomes are "1" (not "0"). Let $\hat{q} = \frac{t}{C}$ (we know $\hat{q} \approx \sin^2\theta$)

0c Compute $\hat{\theta} = \arcsin\sqrt{\hat{q}}$. Digits 1 and 2 are accurate, digit 3 is pretty accurate.

1 Do Step 0a-c again with R^{10}. Digits 2 and 3 are accurate, digit 4 is pretty accurate.

2 Do with R^{10^2}. Digit 3 and 4 are accurate, digit 5 is pretty accurate,

⋮ etc.

Annoying "glitch" angles: $10\theta, 100\theta, 10^3\theta$ might not be between 0° and 90°. In each stage might have to artificially add offset of $45°, 90°, 135°, \ldots, 315°$

Bird's-eye view:

0 Make new qubits $D_1, D_2, \ldots, D_C$, apply R to each, extract $D_1, \ldots, D_C$ and record outcomes into data bits $d_1, \ldots, d_C$

1 Make new qubits $D_{C+1}, \ldots, D_{2C}$, apply R^{10} to each, extract D, get data bits $d_{C+1}, \ldots, d_{2C}$

⋮ continue, making C new qubits each time then applying R^{10^k}, extracting, then recording outcomes into data bits.

End: Simple classical data processing algorithm takes all $d_1, \ldots, d_{Cn}$ and computes estimate of θ.

Takeaway: We can rearrange by doing all new qubits together, doing unitaries, then extracting all. Rotation Estimation can look like this instead:

1. new qubits $D_1 \ldots D_{C \cdot n}$
2. Use unitaries $R, R^{10}, R^{100}, \ldots$

   can handle offsets by just doing all potential offsets here
3. extract $D_1 \ldots D_{C \cdot n}$
4. classically process $d_1 \ldots d_{C \cdot n}$

   as it processes, learns correct offset - just pulls the correct outcome, available from step 2

80.1 Exercises

Let U be a (real) D-dimensional unitary and let $|f_0\rangle$ a unit length D-dimensional vector. Let $|f_1\rangle = U|f_0\rangle$ and assume that $|f_1\rangle$ is not parallel to $|f_0\rangle$ (in other words, $|f_1\rangle \neq \pm|f_0\rangle$). Thus $|f_0\rangle$ and $|f_1\rangle$ span some two dimensional plane P.

Let $|f_2\rangle = U|f_1\rangle = UU|f_0\rangle$. Suppose we somehow "notice" that $|f_2\rangle$ is a linear combination of $|f_0\rangle$ and $|f_1\rangle$, say $|f_2\rangle = a|f_0\rangle + b|f_1\rangle$ for scalars $a, b \in \mathbb{R}$. In other words, we notice that $|f_2\rangle$ is in the plane P.

Show that if $|g\rangle$ is any vector in P, then $U|g\rangle$ is also in P. Conclude that also $|f_k\rangle$ is in P for all $k \geq 0$, where $|f_k\rangle$ is the result of applying U, k times, to $|f_0\rangle$.

Remark: This shows that U maps the 2-D plane P onto itself. Since U is unitary, it doesn't change the angles between any two vectors (in P), we conclude U must be a rotation/reflection of P. Thus our "noticing" of $|f_2\rangle$'s property means we've found one of U's 2-D planes of rotation.

80.2 Solutions

Proof. Let $|g\rangle \in P$. Then $\exists x, y \in \mathbb{R}$ such that $g = x|f_0\rangle + y|f_1\rangle$.

Then $U|g\rangle = U(x|f_0\rangle + y|f_1\rangle) = xU|f_0\rangle + yU|f_1\rangle = x|f_1\rangle + y|f_2\rangle$.

But $|f_2\rangle = a|f_0\rangle + b|f_1\rangle$ for some scalars $a, b \in \mathbb{R}$, so

$U|g\rangle = x|f_1\rangle + y(a|f_0\rangle + b|f_1\rangle) = ay|f_0\rangle + by|f_1\rangle + x|f_1\rangle = ay|f_0\rangle + (by + x)|f_1\rangle$. By closure of $\mathbb{R}$ under addition and multiplication, we've shown that $\exists r, s \in \mathbb{R}$ such that $U|g\rangle = r|f_0\rangle + s|f_1\rangle$, namely $r = ay$ and $s = by = x$. It follows that $U|g\rangle$ is in P, since $U|g\rangle$ is a linear combination of the basis vectors of P.

We can use the above result to just conclude that $|f_k\rangle$ is in P for all $k \geq 0$ by induction.

Base Case: $|f_0\rangle, |f_1\rangle, |f_2\rangle$ are all in P by problem statement.

Induction Step: Assume $|f_k\rangle \in P$ for $k \geq 2$. WTS $|f_{k+1}\rangle \in P$. Well we proved that $\forall |g\rangle \in P$, $U|g\rangle \in P$. Since $|f_k\rangle \in P$ by IH, we know $U|f_k\rangle = |f_{k+1}\rangle$ is also in P, as desired. □

81 Rotation Estimation: different settings

Video #081/100 Table of Contents

Gist: In the simple 1-qubit setting, we knew how to make $|unif\rangle$ and $|f\rangle$. In Grover, the setting was more complicated, where we are able to create $|unif\rangle$, but we don't really know where if f then minus does for $|f\rangle$. So we only really knew how to make one of the vectors in the plane of rotation.

Generalizing the setting: Suppose the given R (and subsequent powers of R) is a unitary acting on many qubits $A_1 \dots A_m$ in initial state $|start\rangle \in \mathbb{R}^{2^m}$, and it's in a 2-D plane of rotation P, where R acts as rotation by θ.

- 1-qubit setting: $m = 1$. We're working in $\mathbb{R}^{2^1} = \mathbb{R}^2$, so we know the rotation plane P. So we can make as many copies of $|start\rangle = |0\rangle$ as we want.
- Grover setting: m is large. R is a combination of the instructions if f then minus and reflect across $|unif\rangle$. P is unknown, but at least we know how to create $|start\rangle = |unif\rangle$, so can still make many copies of the starting state.

  Can still do Rotation Estimation, Can still estimate the angle α that some state vector in $|v\rangle \in P$ makes with $|unif\rangle$ by "undoing" the creation of $|unif\rangle$ by doing Hadamard all (bringing $|unif\rangle \to |00\dots0\rangle$) and $|v\rangle$ to something with angle α with $|00\dots0\rangle$, then extract all and treating the outcome of "00...0" as "0" and anything outer state outcome as "1". The probability of seeing all zeroes is $\cos^2\alpha$ and the probability of not seeing all zeroes is $\sin^2\alpha$.
- Factoring setting (hardest setting): P is unknown, qubits in $A_1 \dots A_m$ are in some state $|start\rangle \in P$, but we don't know how they got there! So we can't just make a bunch of copies and then probabilistically run the Rotation Estimation from Grover and find θ to within 1%.

  But how don't we know how to make $|start\rangle$? In Factoring, we have some complicated R, has 2-D planes of rotation $P_1, P_2, \dots P_N$ whre it rotates by $\theta_1, \theta_2, \dots \theta_N$. It'll be though as if we have the ability to make a random $|start\rangle$ in a random P_j for $1 \le j \le N$. So we can't really make a ton of copies of a known $|start\rangle$.

  Summary of the problem: given state vector $|v\rangle \in P$, we don't know how to measure it against $|start\rangle$ (since we don't know how $|start\rangle$ was made, we can't "undo" the creation of $|start\rangle$).

  How do we solve this problem? The **Hadamard Test**.

81.1 Exercises: Swap Test

The "Swap Test" is a component of quantum state learning algorithms, used to try to check if two qubits are "similar". Suppose you have two unentangled qubits B C in states $|v\rangle, |w\rangle \in \mathbb{R}^2$ respectively. Performing the "Swap Test" on them refers to the following code:

1. new qubit A
2. Add&Diff A
3. if A then swap B C
4. Avg&Dev A
5. extract A

6. if display is "0" output yes, else output no.

Show that the probability of outputting yes is $\frac{1}{2} + \frac{1}{2}\langle v|w\rangle^2$. In particular, the probability is 1 is the qubits are identical and $\frac{1}{2}$ if they are perpendicular.

81.2 Solutions

Proof. We can express our joint state using tensor products.

$$
\begin{aligned}
&(|0\rangle) \otimes (v_0|0\rangle + v_1|1\rangle) \otimes (w_0|0\rangle \otimes w_1|1\rangle) && \text{(Step 1)}\\
\mapsto &(|0\rangle + |1\rangle) \otimes (v_0|0\rangle + v_1|1\rangle) \otimes (w_0|0\rangle \otimes w_1|1\rangle) && \text{(Step 2)}\\
&= (v_0|00\rangle + v_1|01\rangle + v_0|10\rangle + v_1|11\rangle) \otimes (w_0|0\rangle \otimes w_1|1\rangle)\\
&= v_0w_0|000\rangle + v_0w_1|001\rangle + v_1w_0|010\rangle + v_1w_1|011\rangle + v_0w_0|100\rangle + v_0w_1|101\rangle + v_1w_0|110\rangle + v_1w_1|111\rangle\\
\mapsto &v_0w_0|000\rangle + v_0w_1|001\rangle + v_1w_0|010\rangle + v_1w_1|011\rangle + v_0w_0|100\rangle + v_0w_1|110\rangle + v_1w_0|101\rangle + v_1w_1|111\rangle && \text{(Step 3)}\\
\mapsto &(v_0w_0 + v_0w_0)|000\rangle + (v_0w_0 - v_0w_0)|100\rangle\\
&+(v_0w_1 + v_1w_0)|001\rangle + (v_0w_1 - v_1w_0)|101\rangle\\
&+(v_1w_0 + v_0w_1)|010\rangle + (v_1w_0 - v_0w_1)|110\rangle\\
&+(v_1w_1 + v_1w_1)|011\rangle + (v_1w_1 - v_1w_1)|111\rangle\\
&= 2v_0w_0|000\rangle + (v_0w_1 + v_1w_0)|001\rangle + (v_1w_0 + v_0w_1)|010\rangle\\
&+2v_1w_1|011\rangle + (v_0w_1 - v_1w_0)|101\rangle + (v_1w_0 - v_0w_1)|110\rangle && \text{(Step 4.a)}\\
&= v_0w_0|000\rangle + \frac{(v_0w_1 + v_1w_0)}{2}|001\rangle + \frac{(v_1w_0 + v_0w_1)}{2}|010\rangle\\
&+v_1w_1|011\rangle + \frac{(v_0w_1 - v_1w_0)}{2}|101\rangle + \frac{(v_1w_0 - v_0w_1)}{2}|110\rangle && \text{(Step 4.b, divide by 2 to for Avg\&Dev)}
\end{aligned}
$$

Now in Step 5,

$$
\begin{aligned}
\mathbb{P}(\text{"see A outcome 0"}) &= (v_0w_0)^2 + \left(\frac{(v_0w_1 + v_1w_0)}{2}\right)^2 + \left(\frac{(v_1w_0 + v_0w_1)}{2}\right)^2 + (v_1w_1)^2\\
&= (v_0w_0)^2 + \frac{1}{2}(v_0w_1 + v_1w_0)^2 + (v_1w_1)^2\\
&= v_0^2w_0^2 + v_1^2w_1^2 + \frac{1}{2}(v_0^2w_1^2 + 2v_0v_1w_0w_1 + v_1^2w_0^2)\\
&= \frac{1}{2}(v_0^2w_0^2 + v_1^2w_1^2) + \frac{1}{2}(v_0^2w_0^2 + v_1^2w_1^2) + \frac{1}{2}(v_0^2w_1^2 + v_1^2w_0^2) + \frac{1}{2}(2v_0v_1w_0w_1)\\
&= \frac{1}{2}(v_0^2w_0^2 + v_1^2w_1^2 + v_0^2w_1^2 + v_1^2w_0^2) + \frac{1}{2}(v_0^2w_0^2 + v_1^2w_1^2 + 2v_0v_1w_0w_1)\\
&= \frac{1}{2}\left(v_0^2(w_0^2 + w_1^2) + v_1^2(w_1^2 + w_0^2)\right) + \frac{1}{2}(v_0^2w_0^2 + v_1^2w_1^2 + 2v_0v_1w_0w_1)\\
&= \frac{1}{2}\left((v_0^2 + v_1^2)\cdot(w_0^2 + w_1^2)\right) + \frac{1}{2}(v_0^2w_0^2 + v_1^2w_1^2 + 2v_0v_1w_0w_1)\\
&= \frac{1}{2}\left((1)\cdot(1)\right) + \frac{1}{2}(v_0^2w_0^2 + v_1^2w_1^2 + 2v_0v_1w_0w_1)\\
&= \frac{1}{2} + \frac{1}{2}(v_0w_0 + v_1w_1)^2\\
&= \frac{1}{2} + \frac{1}{2}\langle v|w\rangle^2, \text{ as desired.}
\end{aligned}
$$

□

82 The Hadamard Test

Video #082/100 Table of Contents

Nature of the Hadamard Test:

- Inputs:
  - code for R, an m-qubit unitary
  - qubits $A_1 \dots A_m$ in state $|start\rangle$ promsied to be in some plane P that R rotates by θ
- Outputs:
  - a random bit d which is "1" with probability $\sin^2 \frac{\theta}{2}$
  - qubits $A_1 \dots A_m$ are in a new state $|new\rangle$, and $|new\rangle \in P$.

In hardest setting, recall that IntervalEstimation just needs to be repeatedly get random bits d which are "1" with probability $\sin^2 \theta$. After the first Hadamard Test on $|start\rangle$, we can do another one, and keep going, since the state $|new\rangle$ is in P. So $|new\rangle$ becomes the new $|start\rangle$ for the next test.

`HadamardTest` (inputs/outputs specified above)

- new qubit D
- Add&Diff D — superposition state
- if D then do R on $A_1 \dots A_n$ — we have code for R so it's okay
- Avg&Dev D
- d = extract D

How do we do Step 3, which is "controlled R"? For example, say $R(A_1, A_2, A_3)$ is the subroutine

1. toggle A_1
2. if A_2 then toggle A_3
3. Hadamard A_2
4. if A_1 and A_2 then toggle A_3

You basically just prepend "if D then" to all of the lines. So it's like "if D then $R(A_1, A_2, A_3)$"

1. if D then toggle A_1
2. if D AND A_2 then toggle A_3
3. if D then Hadamard A_2
4. if D and A_1 and A_2 then toggle A_3

not a basic instruction, but makeable subroutine in ≈ 6 lines

82.1 Exercises: Incr5

Consider the 3-qubit classical reversible operation Incr5 ("increment mod 5") which maps

- $|000\rangle \mapsto |001\rangle$
- $|001\rangle \mapsto |010\rangle$
- $|010\rangle \mapsto |011\rangle$

- $|011\rangle \mapsto |100\rangle$
- $|100\rangle \mapsto |000\rangle$
- $|101\rangle \mapsto |101\rangle$ no-op
- $|110\rangle \mapsto |110\rangle$ no-op
- $|111\rangle \mapsto |111\rangle$ no-op

This is a real unitary transformation in 8 dimensions, and hence it has some kind of decomposition into 2-D rotations, 1-D reflections, and 1-D no-ops, as per Video #038/100. Of course, in the three dimensions $|101\rangle, |110\rangle, |111\rangle$ it is doing a no-op, so let's just forget about these last three dimensions and think of Incr5 as being the following 5-dimensional real unitary:

$$\begin{bmatrix} v \\ w \\ x \\ y \\ z \end{bmatrix} \mapsto \begin{bmatrix} z \\ v \\ w \\ x \\ y \end{bmatrix}$$

(a) Find one more axis/dimension/unit-vector along which Incr5 is doing a no-op.

(b) This leaves 4 remaining dimensions. I promise you that these split up into two perpendicular 2-D planes P_1 and P_2 in which Incr5 is doing rotations by θ_1 and θ_2 satisfying $0 < \theta_1 < \theta_2 < 180°$. Given that doing Incr5 five times in a row is equivalent to doing nothing, deduce what θ_1, θ_2 must be.

82.2 Solutions

(a) If we pass in the unit vector $(v, w, x, y, z) = \frac{1}{\sqrt{5}}(1, 1, 1, 1, 1)$, Incr5 does nothing. That's the unit-vector along which Incr5 is doing a no-op (it's a unit eigenvector for Incr5's path matrix).

(b) $\frac{\tau}{5} = 72°$. The additional constraints from the problem statement indicate that $\theta_1 = 72°$ and $\theta_2 = 144°$.

83 The Hadamard Test: analysis

Video #083/100 Table of Contents

`HadamardTest` (inputs/outputs specified in Video #082/100)

- new qubit D

  After line 1, joint state $|start\rangle \otimes |0\rangle$.

- Add&Diff D superposition state

  After line 2, joint state $|start\rangle \otimes (|0\rangle + |1\rangle) = |start\rangle \otimes |0\rangle + |start\rangle \otimes |1\rangle$.

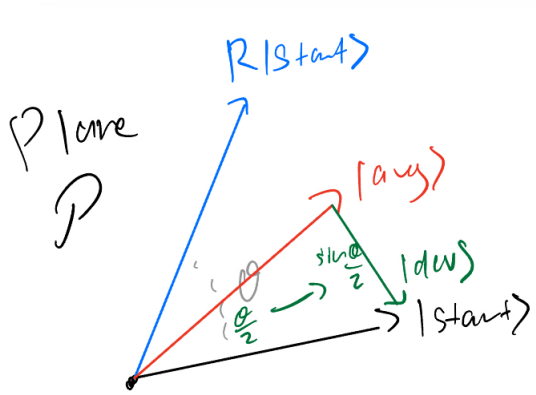
- if D then do R on $A_1 \dots A_n$ we have code for R so it's okay

  After line 3, $|start\rangle \otimes |0\rangle + (R\,|start\rangle) \otimes |1\rangle$. It's as if you both did and didn't do R on start, and you're trying to compare the amplitudes for both cases.

- Avg&Dev D

  After line 4, $|avg\rangle \otimes 0 + |dev\rangle\,|1\rangle$ where $|avg\rangle = \frac{|start\rangle + R|start\rangle}{2}$ and $|dev\rangle = \frac{|start\rangle - R|start\rangle}{2}$.

  $|start\rangle$ and $R\,|start\rangle$ are in a 2-D plane P and make angle θ. $|avg\rangle$ and $|dev\rangle$ are also both in this plane P.



- d = extract D

  $\mathbb{P}(d = 0) = \langle avg|avg\rangle = len(|avg\rangle)^2 = \cos^2\frac{\theta}{2}$ and if this occurs, new state of $A_1 \dots A_m$ is $|avg\rangle$ (normalized)

  $\mathbb{P}(d = 1) = \langle dev|dev\rangle = len(|dev\rangle)^2 = \sin^2\frac{\theta}{2}$ and if this occurs, new state of $A_1 \dots A_m$ is $|dev\rangle$ (normalized)

83.0.1 Final Form of Rotation Estimation

- Inputs:
  - n, the # of digits of precision you want
  - code for unitary R on $A_1 \dots A_n$ and also code for $R^{10}, R^{100}, \dots, R^{10^n}$
  - $A_1 \dots A_m$ initialized to state $|start\rangle$ in one of R's 2-D planes of rotation
- Form:

- for count C, makes new qubits $D_1 \dots D_{C \cdot n}$
- uses the code for $R, R^{10}, \dots, R^{10^n}$ C times each.
- Extracts all D_i's, giving classical bits $d_1 \dots d_{C \cdot n}$
- Simple classical algorithm processing d_i's gives final estimate $\hat{\theta}$

- Output: *wowp*, $\hat{theta} \approx \theta$ to n digits of precision

Boring fine print:

Q: Don't we need to assume $0 \leq \theta \leq 90°$?

A: Actually, since Hadamard Test involves $\frac{\theta}{2}$ and not θ, it suffices to assume $0 \leq \theta \leq 180°$. And actually every angle is in this range if you don't differentiate $\pm\theta$ (and it doesn't really make sense to try to differentiate them anyway)

Q: What about the glitch? Doesn't the algorithm also need to know how to do rotations by multiples of $45°$ within P?

A: According to the way we explained it, yes it does (Well actually, because of the $\frac{\theta}{2}$ vs. θ thing, it only needs to be able to do rotations by multiples of $90°$, but that doesn't help much.) There are two answers:

1. With more annoying details, it's possible to get around this. We won't get into it
2. When we actually use Rotation Estimation in the Factoring algorithm, it'll turn out that we actually know how to do rotations by any angle we want within P. I'll point this out when it arises.

83.1 Exercises (continued from last one)

(c) Geometric interlude: Suppose U is a (real) d-dimensional unitary, and $|f\rangle$ is a (nonzero) d-dimensional vector. Moreover, suppose you notice that if we define $|f'\rangle := U|f\rangle$, and $|f''\rangle := UU|f\rangle$, then $|f\rangle + |f''\rangle = c|f'\rangle$ for some scalar c (say, with $c \neq \pm 1$). It actually follows that $|f\rangle$ must lie in one of u's 2-D planes of rotation. Say the angle U rotates by in this plane is θ. Draw a picture illustrating $|f\rangle, |f'\rangle, |f''\rangle$ and explain how c is related to θ (For this, you may assume $|f\rangle$ is a unit vector).

(d) Back to part (b). Actually finding two vectors $|f_1\rangle, |g_1\rangle$ that span P_1 and two vectors $|f_2\rangle, |g_2\rangle$ that span P_2 is kind of annoying (unless you like to use complex numbers, in which case it's super easy! Sometimes complex numbers make things simpler, not more complex!) Yet it will be extremely important for the quantum factoring algorithm to understand these planes of rotation for the general IncrL ("increment mod L") operation, so later I will show you One Weird Trick that makes it not annoying. But for now I will annoy you, just so you feel less annoyed when you see the One Weird Trick.

Let ϕ stand for a number satisfying $\phi^2 + \phi - 1 = 0$, and consider the vector $|f\rangle = \begin{bmatrix} \phi \\ 1+\phi \\ 1-\phi \\ \phi-2 \\ -2\phi \end{bmatrix}$. Show that we are in the scenario of part (c), with $c = \phi$.

(e) By using the quadratic formula, identify the two possible values for ϕ as $\varphi i - 1$ and $-\varphi$, where φ is the golden ratio.

83.2 Solutions

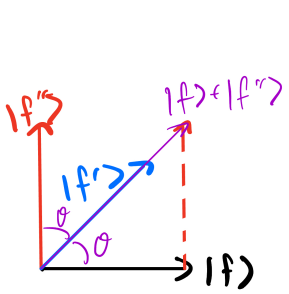
(c) $c = 2\cos\theta$.

Assume U rotates by angle θ. Then $\left|f'\right\rangle = \cos\theta\left|f\right\rangle + \sin\theta\left|f^{\perp}\right\rangle$ and $\left|f''\right\rangle = \cos 2\theta\left|f\right\rangle + \sin 2\theta\left|f^{\perp}\right\rangle$, where $\left|f^{\perp}\right\rangle$ is $Rot_{90^\circ}\left|f\right\rangle$ in the plane that contains $\left|f\right\rangle, \left|f'\right\rangle, \left|f''\right\rangle$. WTS $c\left|f'\right\rangle = \left|f\right\rangle + \left|f''\right\rangle$.

$$\begin{aligned}
\left|f\right\rangle + \left|f''\right\rangle &= \left|f\right\rangle + \cos 2\theta\left|f\right\rangle + \sin 2\theta\left|f^{\perp}\right\rangle \\
&= (1+\cos 2\theta)\left|f\right\rangle + (\sin 2\theta)\left|f^{\perp}\right\rangle \\
&= (2\cos^2\theta)\left|f\right\rangle + (2\sin\theta\cos\theta)\left|f^{\perp}\right\rangle \\
&= 2\cos\theta(\cos\theta\left|f\right\rangle + \sin\theta\left|f^{\perp}\right\rangle) \\
&= 2\cos\theta(\left|f'\right\rangle)
\end{aligned}$$

Therefore, $c = 2\cos\theta$.

Example: let $\left|f\right\rangle = \left|0\right\rangle$, and let $U = Rot_\theta$. WTS $cRot_\theta\left|0\right\rangle = \left|0\right\rangle + Rot_{2\theta}\left|0\right\rangle$. $c\begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}\cos 2\theta\\ \sin 2\theta\end{bmatrix}$. Then $c\cos\theta = 1 + \cos 2\theta = 1 + \cos^2\theta - \sin^2\theta = 2\cos^2\theta$ and $c\sin\theta = \sin 2\theta = 2\sin\theta\cos\theta$. Both expressions hold true when $c = 2\cos\theta$. And in the drawing, where $\theta = 45^\circ$, $2\cos\theta = \sqrt{2} \approx 1.4$, which is about the scaling factor to get from the blue vector to the purple vector.



(d) Let U be Incr5. Then

$$\left|f\right\rangle = \begin{bmatrix}\phi\\1+\phi\\1-\phi\\\phi-2\\-2\phi\end{bmatrix}, \left|f'\right\rangle = \begin{bmatrix}-2\phi\\\phi\\1+\phi\\1-\phi\\\phi-2\end{bmatrix}, \left|f''\right\rangle = \begin{bmatrix}\phi-2\\-2\phi\\\phi\\1+\phi\\1-\phi\end{bmatrix} \implies \left|f\right\rangle + \left|f''\right\rangle = \begin{bmatrix}2\phi-2\\1-\phi\\1\\2\phi-1\\1-3\phi\end{bmatrix} \text{ and } \phi\left|f'\right\rangle = \begin{bmatrix}-2\phi^2\\\phi^2\\\phi+\phi^2\\\phi-\phi^2\\\phi^2-2\phi\end{bmatrix}$$

- $-2\phi^2 = 2\phi - 2 \iff 0 = 2\phi^2 + 2\phi - 2 \iff \phi^2 + \phi - 1 = 0$ ✓
- $\phi^2 = 1 - \phi \iff \phi^2 + \phi - 1 = 0$ ✓
- $\phi + \phi^2 = 1 \iff \phi^2 + \phi - 1 = 0$ ✓
- $\phi - \phi^2 = 2\phi - 1 \iff 0 = \phi^2 - \phi + 2\phi - 1 \iff \phi^2 + \phi - 1 = 0$ ✓
- $\phi^2 - 2\phi = 1 - 3\phi \iff \phi^2 - 2\phi + 3\phi - 1 = 0 \iff \phi^2 + \phi - 1 = 0$ ✓

So we are in the scenario from part (c).

(e) $\phi = \frac{-1\pm\sqrt{1^2-4(1)(-1)}}{2(1)} = \frac{-1\pm\sqrt{1+4}}{2} = \frac{-1\pm\sqrt{5}}{2}$. Recall $\varphi = \frac{1+\sqrt{5}}{2}$. Then $-\varphi = \frac{-1-\sqrt{5}}{2}$ and $\varphi - 1 = \frac{1-2+\sqrt{5}}{2} = \frac{-1+\sqrt{5}}{2}$. So $\varphi - 1, \varphi = \frac{-1\pm\sqrt{5}}{2}$ respectively.

84 Factoring Algorithm: the overview (Lecture 22)

Video #084/100 Table of Contents

Factoring large positive integers. With a quantum computer, can factor n-digit numbers in $O(n^3)$ steps. Shor's Algorithm (1994) uses quantum Fourier transform, but we're going to go over Kitaev's algorithm instead (though they are essentially the same).

Goal: given N (think thousands of digits), output its prime factorization. Simplifying assumption: $N = P \cdot Q$.

Overview of the algorithm

1. Find the smallest number L such that $2^L \equiv 1 \bmod N$ — this is the hard and quantum step
2. Let $X = 2^{\frac{L}{2}} \bmod N$ — (modular exponentiation, is efficient)
3. Output $\gcd(X-1, N)$ and $\gcd(X+1, N)$ — (Euclidean algorithm, is efficient)

Major Issue that we need quantum for is Step 1.

E.g., $N = 15$. Then we check powers of 2. $2, 4, 8, 16 \equiv 1 \bmod 15$! So we set $L = 4$, $X = 2^{\frac{4}{2}} = 4$. $\gcd(4-1, 15) = \gcd(3, 15) = 3$. $\gcd(4+1, 15) = \gcd(5, 15) = 5$. So $15 = 3 \cdot 5$.

Minor issue 1: What if L turns out to not be even?

Minor issue 2: Why does this algorithm work?

Proof. $X^2 = \left(2^{\frac{L}{2}}\right)^2 = 2^L \equiv 1 \bmod N$. Then $X^2 - 1 \equiv 0 \bmod N$. Using difference of squares, $(X+1) \cdot (X-1) \equiv 0 \bmod N$. Then we know $N \mid (X+1) \cdot (X-1)$, so $(PQ) \mid (X+1) \cdot (X-1)$ Then P is a prime factor of exactly one of $X-1, X+1$. Same for Q. □

But if the minor issues don't happen, P divides $X-1$ but not $X+1$ and Q divides the remaining one. Then $\gcd(X-1, N) = P$ and $\gcd(X+1, N) = Q$. In a later lecture, we will see that both these minor issues are fixed by doing the algorithm with "bases" B other than $B = 2$ ("base" for X)

84.1 Exercises

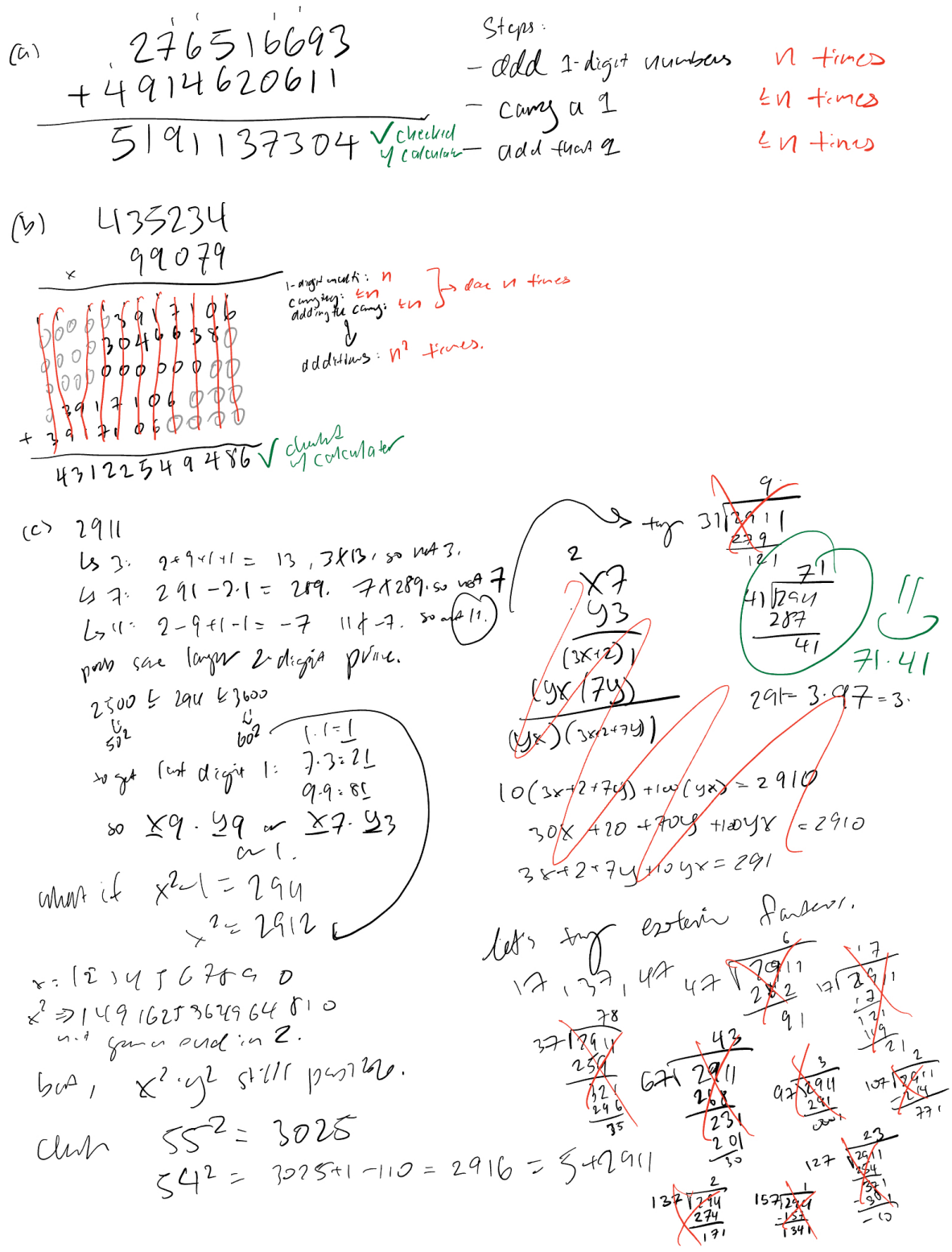
I find that a great way to reason about the $O(\cdot)$ time of an algorithm is to think about how many steps it would take *YOU*, literally **YOU**, to do the algorithm with lots of paper and pencils and coffee, when $n = 10$ or 20. Then imagine a computer doing the same routine when $n = 10$ billion or 20 billion.

(a) Add the numbers 276516693 and 4914620611. Like literally do it, by hand, on paper. What is the answer? Do you get the sense why "adding two n-digit numbers" can be done in $O(n)$ steps?

(b) Multiply the numbers 435234 and 99079. Again, literally do it, for realsies. What is the answer? How many "basic instructions" do you feel you did? Do you get the sense why "multiplying two n-digit numbers" can be done in $O(n^2)$ steps? (Pretty nifty that it can also be done in $O(n)$ steps!)

(c) Find the prime factorization of 2911, by hand. Show your work. As a human, do you feel you have any hope of factoring this 10-digit number: 5126822089? Why or why not?

84.2 Solutions



Absolutely no way I want to factor a 10-digit number. There are some things that make it kind of doable: for example, the final digit 9 means two factors are going to have units digits (1,9) or (3,3) or (7,7). But that's about all we get.

85 We need to find the length of a cycle

Video #085/100 Table of Contents

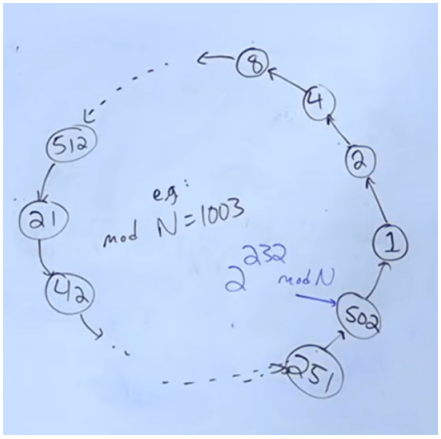
Bad idea: try to bash out L.

1. $L := 1$, $v := 2$.
2. loop until $v = 1$:

   $L+ = 1$

   $v =$Times2ModN(v)

Essentially, we are trying to find the length of a cycle. Luckily here, we only needed about $\frac{N}{4}$ iterations. But that's neither guaranteed to always happen nor very efficient anyways.



This is a graph, more specifically a directed cycle, where very vertex has out degree 1 and in degree 1. We can find the predecessor by multiplying by the multiplicative inverse of 2 in mod 1003 (which is 502). Note that this is cycle will contain exactly L numbers mod N; will not guarantee all numbers in mod N. We only care about the cycle that contains 1.

Summary: given N, the powers of 2 mod N for a set S, which has L elements, naturally arranged on a cycle. To factor N, we want to find L. We have efficient classical code for Times2ModN, which (cyclically) permutes elements of S. We also have the inverse permutation Times1/2ModN. Hence, we can convert the efficient classical code for those two operations to efficient quantum code Times2ModN on qubits. The unitary for Times2ModN is the adjacency matrix for the directed cycle.

85.1 Exercises

Consider the long division algorithm for integers that you learn in grade school. Given two numbers C and D, it outputs the (integer) quotient $Q = \lfloor C/D \rfloor$ and the remainder $R = C \bmod D$. Briefly explain why, if C and D are both at most n digits, this algorithm will compute Q and R in at most $\tilde{O}(n^2)$ operations.

Remark: in fact, there's a sophisticated way to reduce integer divsion to integer multiplication, meaning that integer division can actually be done in $\tilde{O}(n)$ operations. The infamous Pentium bug was due to messing up this reduction.

85.2 Solutions

Each step of long division has several parts:

- Start with the first n_C digits of D (indexed from left 1 to right $\leq n$).
- Find a multiple of C that divides $D[1 : n_C]$ (using Pythonic array slicing notation)
- Find $D[1 : n_C] \bmod C$. This is the remainder.
- If there are still digits left in D, multiply the remainder by 10 and add the next digit in D.
- Repeat.

Worst case, C is 1-digit and D is n-digit. Then it'll be about n of these steps, and each step has integer multiplication (which we said in the previous exercises can be done in $O(n)$ steps with some cleverness). So worst case is about $O(n^2)$ steps.

86 Rotation Estimation gives clues about L

Video #086/100 Table of Contents

Let's write R for the unitary operation associated to Times2ModN. We want to do Rotation Estimation on R. We're going to try to get $\frac{1}{L}$ out of this, so we'll need a lot of digits of precision.

We have efficient code for R. Now we're going to make efficient code for $R^{10}, R^{100}, \ldots, R^{10^n}$. R^{10^k} is just repeat R, 10^k times. This is the same as doing $\times 2$, 10^k times in mod N. But this is just multiplying by 2^{10^k} mod N, which can be computed efficiently with modular exponentiation. So basically, when we do Rotation Estimation, we can get as many digits of precision as we want.

n-digit number requires $2n$ digits of precision (will be showed later).

Crucial Fact: R is unitary. The 2-D planes of rotation associated to R have rotation angles $\frac{i}{L}\tau$, where $0 \leq i \leq L-1$. These are the "symmetries" of the directed cycle graph, so they're also the rotation angles for R.

But how will we get a starting state in any of these planes of rotation?

Another Fact: if we do Rotation Estimation with qubits in state $|start\rangle$ (where start is the binary string corresponding to 1 in base 10). we'll get a random one of those angles from the Crucial Fact.

Conclusion: there's an efficient quantum algorithm that prints out (wowp) as many digits as we want as desired of $\frac{K}{L}$ where K is a uniform random number between $0 \& L-1$. $\frac{K}{L}$ is a "clue" to find L.

Finding L from the clue is doable because: you just repeat this efficient quantum algorithm and get a bunch of clues. Then there's a classical algorithm that puts these clues together to find L.

86.1 Exercises

Consider the following task: Given positive integers B and C, compute the integer B^C. Explain why this task is not solvable "`in P`", that is, there is no algorithm that can do this in $O(n^{constant})$ operations when B and C are n-bit numbers.

Consider this task: Given positive integers B, C, D, compute the integer $B^C \bmod D$. This is called the modular exponentiation problem, and it is solvable "`in P`", as perhaps you learned in a CS theory course. If B, C, D are all n-bit numbers, show that the task can be done in $O(n^3)$ steps. In fact, it can be done in $\tilde{O}(n^2)$ steps using the sophisticated multiplication and division algorithms.

Hint: One key fact to use is $P \cdot Q \bmod D = (P \bmod D) \cdot (Q \bmod D) \bmod D$. Given this, first think about computing $B \bmod D$, $B^2 \bmod D$, $B^4 \bmod D$, and so on. If C happens to be a power of 2, you should be in good shape. What should you do if C is, say, 13? What should you do if C is (when represented in base 2) 1010101010101010?

86.2 Solutions

How big is B^C? The number of bits required to represent B^C is $\log_2(B^C)$. Using properties of logarithms, this is equivalent to $C \log_2(B)$. Since B is an n bit number, this is equivalent to Cn. But C is also an n-bit number, so C can be up to 2^n (noninclusive) in value. So the number of output bits for B^C can be up to $2^n \cdot n$, which is not "`in P`" (there is no polynomial time that outpaces the exponential term 2^n.)

Compute $B^C \bmod D$ by first computing $B \bmod D$. Then square the result to find $B^2 \bmod D$. Then square the result to find $B^4 \bmod D$. Continue to do so until you reach an exponent that is less than or equal to C. If C is 13, we can solve up to B^8. Then we'd just express B^{13} as $B^8 \cdot B^4 \cdot B^1$. If C is some bitstring, we'd just multiply by the positions where the bitstring is 1. For example, $B^{1010101010101010_2} = B^{1000000000000000_2} \cdot B^{0010000000000000_2} \ldots$ you get the point. In the $C = 13$ example, $13 = 1101_2$, hence why we expressed $C^{1101_2} = C^{1000_2} \cdot C^{0100_2} \cdot C^{0001_2}$.

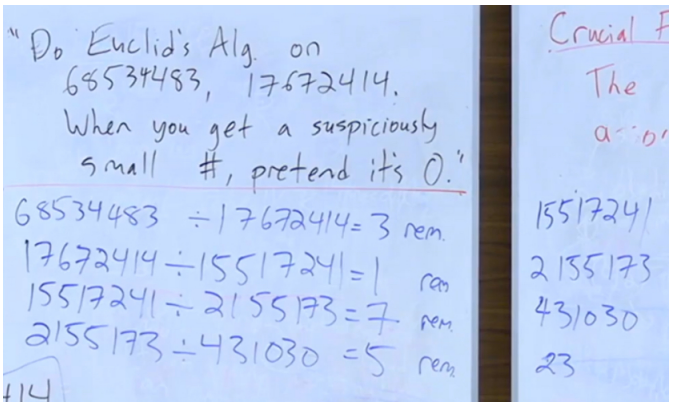
Why is this an $O(n^3)$ algorithm? Well we need at most $\log_2(C)$ steps to make our "building blocks". Since C is an n-bit integer, $\log_2(C) = n$. Then, we have to perform at most n integer multiplications, since C's base 2 representation has at most n 1's.

87 Finding "L" from the clues

Video #087/100 Table of Contents

Number Theory Fact: there's an efficient classical algorithm determining L (wowp) given just a few clues.

Illustration: say N is 4 digits, and we get 8 digits of accuracy from Rotation Estimation. Say we get clues $\frac{K_1}{L} = .68534483$ and $\frac{K_2}{L} = .17672414$. Now we do Euclid's gcd algorithm on the two clues without the decimal point. When you get a suspiciously small number, pretend it's 0.



Just pretend 23 is 0. Then 2155173 divided by 431030 is about 5. Now set $g = 431030$. It follows that 15517241 is about $36g$, and eventually, you'll backpropagate to find that 68534483 is about $159g$. Then $\frac{\frac{K_2}{L}}{\frac{K_1}{L}} \approx \frac{159g}{41g} = \frac{159}{41} = \frac{K_2}{K_1}$. So $L \approx \frac{K_2}{\frac{K_2}{L}} = \frac{159}{.65534483} \approx \frac{K_1}{\frac{K_1}{L}} = \frac{41}{0.17672414}$

87.1 Exercises

This problem is about the task of computing the GCD (greatest common divisor) of two input positive integers, A and B. For the sake of intuition, you should imagine that these numbers are thousands of bits long. The grade school algorithm for GCD is: (i) find the prime factorizations of A and B; (ii) pick out all the common prime factors, and multiply them together. Of course, this algorithm is not actually possible to implement in physical reality for 1000-bit numbers (given known classical factoring algorithms). However, there is a physically possible algorithm (i.e., computing GCD is in "P"): it is called Euclid's Algorithm, and is arguable the first known nontrivial algorithm in history.

(a) Warmup to Euclid's Algorithm: how he actually described it. Very briefly explain why each of the following facts are true:

- if Q is a divisor of both A and B, then it's also a divisor of $A - B$
- if Q is a divisor of $A - B$ and B, then it's also a divisor of A
- The rule $GCD(A, B) = GCD(A - B, B)$ holds.

(b) Warmup continuation. Of course, we also have the rule $GCD(A, B) = GCD(B, A)$. By iterating these two rules, compute $GCD(42, 30)$ by hand. The base case is that $GCD(A, 0) = A$ for all A (since everything is a divisor of 0). You should have to do 5 subtractions in total.

87.2 Solutions

I'm very glad I'm self-studying this right after taking 15-151. Lightwork.

(a) Proving the bulletpoints:

- Prove $Q \mid A \wedge Q \mid B \implies Q \mid (A - B)$. Assume Q divides A and Q divides B. By definition of divides, there are integers k, j such that $Qk = A$ and $Qj = B$. Now we want to show that there exists integer ℓ such that $Q\ell = (A - B)$. Consider $\ell = k - j$. $Q(k - j) = Qk - Qj = A - B$, as desired.
- Prove $Q \mid (A - B) \wedge Q \mid B \implies Q \mid A$. Assume Q divides $A - B$ and Q divides B. By definition of divides, there are integers ℓ, j such that $Q\ell = A - B$ and $Qj = B$. Now we want to show that there exists integer k such that $Qk = A$. Consider $k = \ell + j$. $Q(\ell + j) = Q\ell + Qj = A - B + B = A$, as desired.
- Let S be the set of divisors for A, B and R be the set of divisors for $A - B, B$. We've shown $Q \mid A \wedge Q \mid B \implies Q \mid (A - B)$, so $S \subseteq R$. We've also shown $Q \mid (A - B) \wedge Q \mid B \implies Q \mid A$, so $R \subseteq S$. By double containment, $R = S$. Since the set of divisors for two positive integers must be finite, it must have a greatest element (I'm not going to prove this, the proof is in Clive, infdesc). Then the greatest divisor in R is the same as the greatest divisor in S. Thus $GCD(A, B) = GCD(A - B, B)$.

(b) $GCD(42, 30) = GCD(42 - 30, 30) = GCD(12, 30) = GCD(12, 30 - 12 \times 2) = GCD(12, 6) = GCD(12 - 6 \times 2, 6) = GCD(0, 6) = GCD(6, 0) = 6$. This is technically 5 subtractions:

1. $42 - 30$
2. $30 - 12 \times 2$ is 2 subtractions
3. $12 - 6 \times 2$ is 2 subtractions

88 Planes of rotation for IncrL, part 1 (Lecture 23)

Video #088/100 Table of Contents

Summary of last lecture:

1. Factoring n-digit number $N = P \cdot Q$.
2. Have quantum code for Times2ModN, which cyclically permutes S, a set of size L (powers of 2 mod N)
3. Let R be associated unitary
4. Rotation Estimation on R wish $|start\rangle = |00\ldots01\rangle$ yields (wowp, $2n$ digit of precision) $\frac{K}{L}\tau$, for a random $K \in \{0, 1, 2, \ldots, L-1\}$
5. Getting a few such $\frac{K}{L}\tau$ values is enough to determine L and thus factor N

Applying R is basically like IncrL, cyclically shifting down. We can just pretend everything is happening in L-dimensional space. Here's the trick from Video #083/100's exercises: it makes the math easier

- Times2ModN operates on qubits $A_1 \ldots A_N$.
- Introduce one new qubit B
- Let R be associated unitary on $A_1 \ldots A_N B$. So, e.g. $|start\rangle \otimes |0\rangle$ just ignores the last qubit and does its usual increment. Thus, $R(|00001\rangle \otimes |w\rangle) = Incr6\,|00001\rangle \otimes |w\rangle = |00010\rangle\,|w\rangle$

88.1 Exercises (continued from last one)

(c) Suppose you were computing $GCD(A, 6)$ where $A = 6 \times 10^{500} + 4$. You would not want to do the subtraction rule 10^{500} times before getting to the swapping rule. But..., it should be obvious what subproblem you'll get down to after performing all those subtractions...

Show that the following is a correct algorithm for computing the GCD:

```
Euclid(A,B):
    if B = 0, return A
    else return Euclid(B, A mod B)
```

(d) When we execute `Euclid(A,B)`, it produces a descending chain of numbers (e.g., Euclid(100,18) produces 100, 18, 10, 8, 2). Any three consecutive numbers in this chain are of the form $C, D, (C \text{ mod } D)$. Establish that for any consecutive numbers F_{t-1}, F_t, F_{t+1} in the chain, we have $F_{t+1} \leq \frac{1}{2}F_{t-1}$.

(Hint, case analysis based on how large F_t is, compared with F_{t-1}). Conclude that $F_t F_{t+1} \leq \frac{1}{2}F_{t-1}F_t$; further conclude that the total length of the chain is at most $\log_2(A) + \log_2(B)$.

Thus if A is m bits long, and B is n bits long, the length of the chain produced by Euclid(A,B) is at most $m + n$. Which is not really that long. Thus Euclid's algorithm is efficient.

88.2 Solutions

(c) Proving Prof. Mackey's "Big Theorem":

WTS that $GCD(A, B) = GCD(A \bmod B, B)$.

- Assume Q divides A and B. By definition of divides, $\exists k, j \in \mathbb{Z}$ such that $Qk = A$ and $Qj = B$. WTS $\exists \ell \in \mathbb{Z}$ such that $Q\ell = A \bmod B$. WLOG, by division theorem, we can write $B = qA + r$, where integers q and r are the quotient and remainder respectively (as defined in Clive, but I don't want to write out the full specifics). Then we are actually trying to show $Q\ell = r$. Consider $\ell = j - kq$. Then $Q\ell = Q(j - kq) = Qj - q(Qk) = B - qA = r$, as desired.
- Assume Q divides $A \bmod B$ and B. By definition of divides, $\exists \ell, j \in \mathbb{Z}$ such that $Q\ell = A \bmod B$ and $Qj = B$. WTS $\exists k \in \mathbb{Z}$ such that $Qk = A$. Consider $k = \frac{j-\ell}{q}$. Then $Qk = Q\left(\frac{j-\ell}{q}\right) = \frac{Qj-Q\ell}{q} = \frac{B-r}{q} = \frac{Aq}{q} = A$, as desired. We know k is an integer because of the specification of q by division theorem, and the specification of $A \bmod B$ as the remainder r.
- We've shown Q divides A and B implies Q divides $A \bmod B$ and B, and we've shown the converse. By double containment, the two sets of divisors are the same, hence $GCD(A, B) = GCD(A \bmod B, B)$ as desired.

(d) Lemma: We know $F_{t+1} < F_t$ since it's a value in the modular "residues" of F_t, AKA F_{t+1} is between 0 and $F_t - 1$ inclusive.

- Case $F_t \leq \frac{1}{2}F_{t-1}$. Then by Lemma $F_{t+1} < F_t$, we know $F_{t+1} < \frac{1}{2}F_{t-1}$.
- Case $F_t > \frac{1}{2}F_{t-1}$. By Lemma $F_t < F_{t-1}$, we know $\frac{F_{t-1}}{F_t} > 1$. But by case assumption, we know $\frac{F_{t-1}}{F_t} < 2$. Therefore, $\lfloor \frac{F_{t-1}}{F_t} \rfloor = 1$.
  Then $F_{t+1} = F_{t-1} \bmod F_t = F_{t-1} - F_t \lfloor \frac{F_{t-1}}{F_t} \rfloor = F_{t-1} - F_t$.
  But $F_t > \frac{1}{2}F_{t-1}$. So $-F_t < -\frac{1}{2}F_{t-1}$. Then $F_{t-1} - F_t < F_{t-1} - \frac{1}{2}F_{t-1}$. Then $F_{t+1} < F_{t-1} - \frac{1}{2}F_{t-1}$. Thus $F_{t+1} \leq \frac{1}{2}F_{t-1}$.

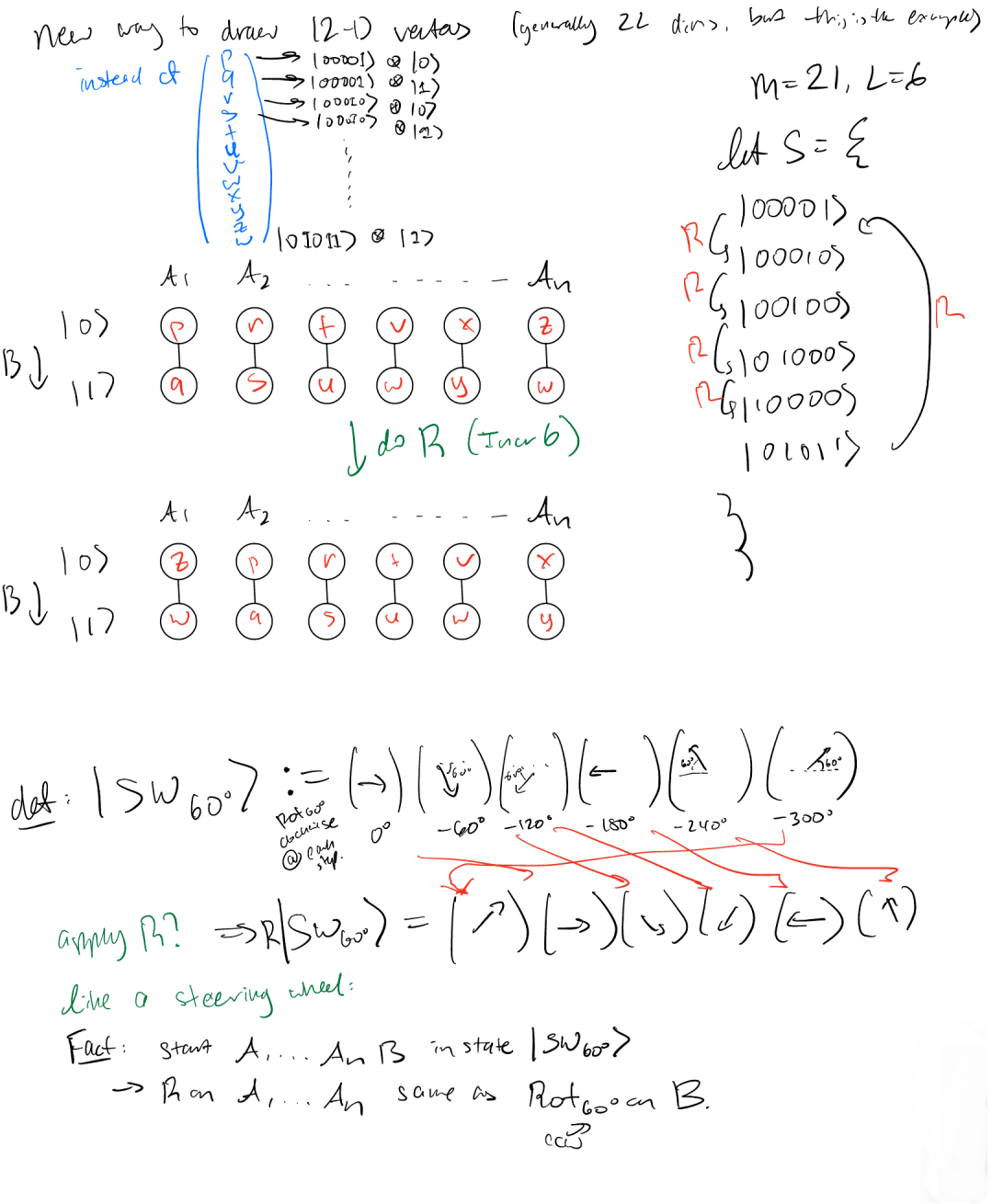
Since $F_{t+1} \leq \frac{1}{2}F_{t-1}$, multiply both sides by F_t (which is positive) to get $F_tF_{t+1} \leq \frac{1}{2}F_{t-1}F_t$.

Then thinking of F_tF_{t+1} as a sliding window of size 2, its value is at least halved for each increment of t. Thus the length of the chain is how many times we can halve starting from the first two elements in the chain, AKA how many times can we halve AB? But then that's just $\log_2(AB) = \log_2(A) + \log_2(B)$ by properties of logarithms.

89 One Steering Wheel vector for IncrL

Video #089/100

Table of Contents



89.1 Exercises

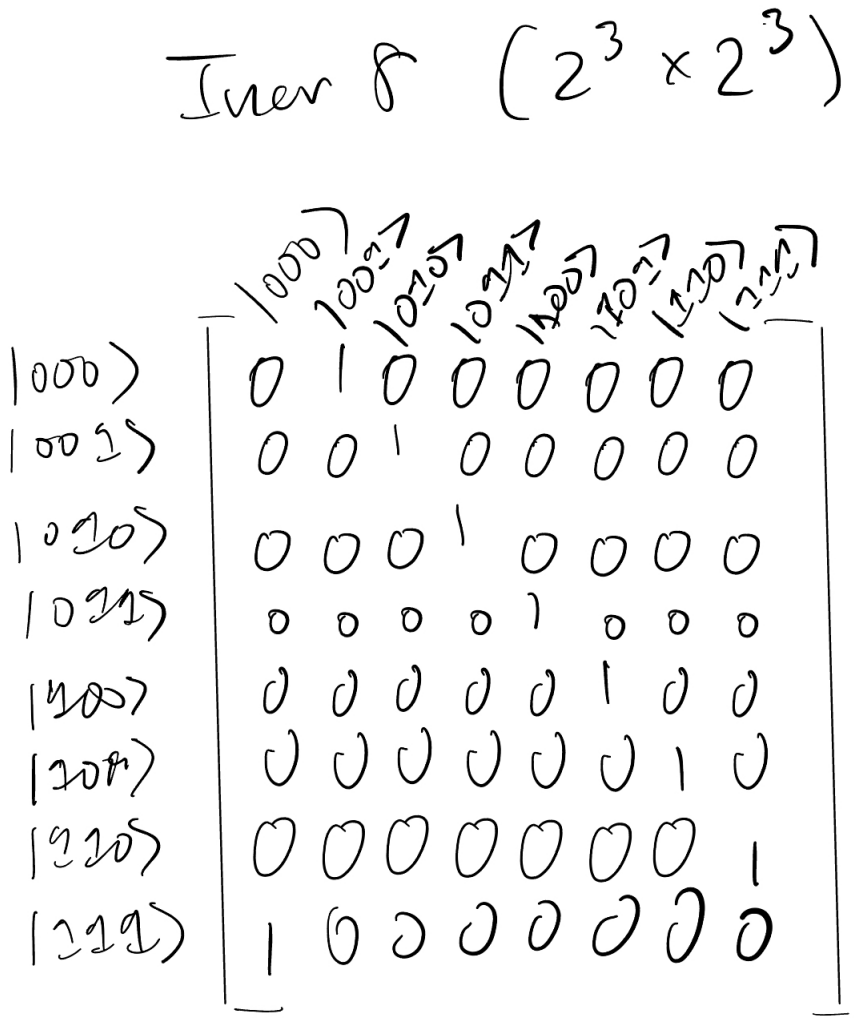
Consider the classical reversible operation Incr8 "Increment Mod 8" which operates on 3 qubits as

$$Incr8\,|000\rangle = |001\rangle\,, Incr8\,|001\rangle = |010\rangle\,, \ldots, Incr8\,|111\rangle = |000\rangle$$

Also, define Incr8' to be the 4-qubit classical reversible operation that does Incr8 on the first 3 qubits and ignores the 4th qubit.

(a) Write the (unitary) matrix representation of Incr8, labeling the rows and columns,

89.2 Solutions

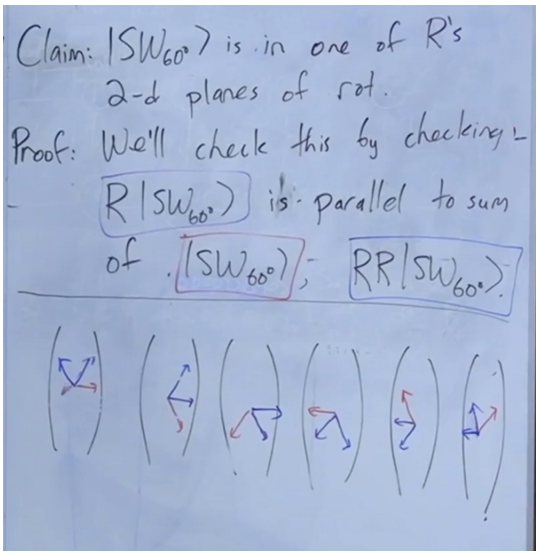


90 Verifying the first SW vector properties

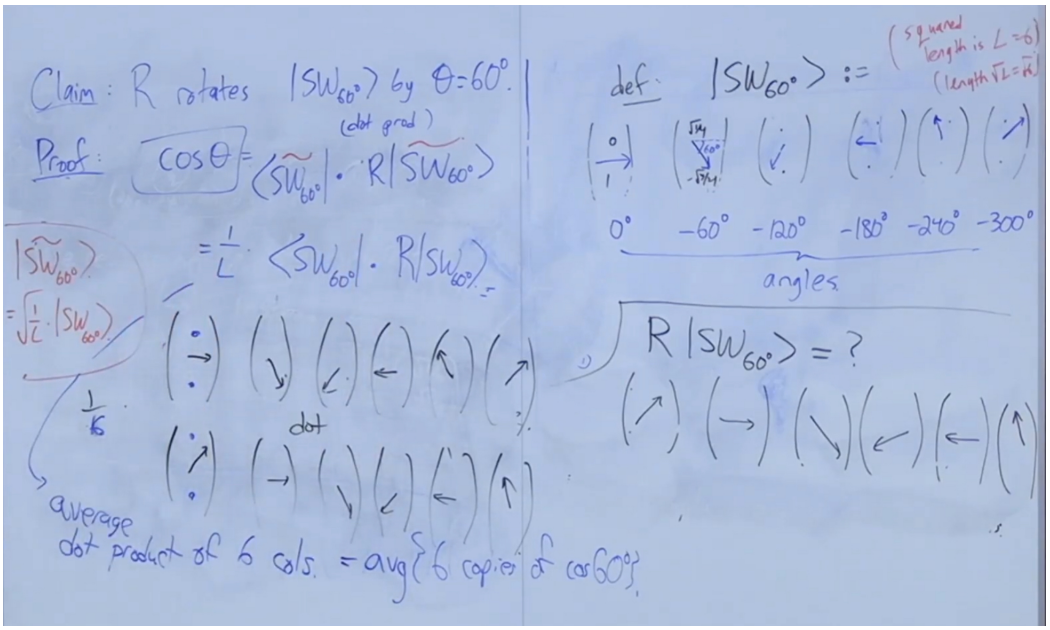
Video #090/100 Table of Contents

Let's check that this vector is in some 2-D plane and when you apply R, you move $|SW_{60^\circ}\rangle$ within this plane.

Claim: $|SW_{60^\circ}\rangle$ is in one of R's 2-D planes of rotation. We'll check this by checking that $R|60^\circ\rangle$ is parallel to the sum of $|60^\circ\rangle$ and $RR|60^\circ\rangle$ (like from exercises for Video #083/100).



Claim: R rotates $|SW_60^\circ\rangle$ and apply R once. What is the angle θ? We know $\cos\theta$ is $\langle SW_{60^\circ}| \cdot R|SW_{60^\circ}\rangle$. But actually, $|SW_{60^\circ}\rangle$ is not a normalized state; the sum of squares of its amplitudes is actually 6. So just normalize this dot product (Prof. adds a twiddle on top to indicate that). So truly, $\cos\theta = \frac{1}{L}\langle SW_{60^\circ}| \cdot R|SW_{60^\circ}\rangle$ = average dot product of the 6 (L) columns. Each of the dot products are all $\cos 60^\circ$, so the average is also $\cos 60^\circ$.



90.1 Exercises (continued from last one)

(b) Write out the associated "steering wheel" vector $|SW_{45°}\rangle$ for Incr8', both in the conventional format of a height-16 column of numbers, and as a row of 8 unit vectors in 2-D.

90.2 Solutions

$$|SW_{45°}\rangle = \begin{pmatrix} 1 \\ 0 \\ \sqrt{2}/2 \\ -\sqrt{2}/2 \\ 0 \\ -1 \\ -\sqrt{2}/2 \\ -1 \\ 0 \\ -\sqrt{2}/2 \\ \sqrt{2}/2 \\ 0 \\ \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix}$$

$|0\rangle$ $|1\rangle$

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

$0°$ $-45°$ $-90°$ $-135°$ $-180°$ $-225°$ $-270°$ $-315°$

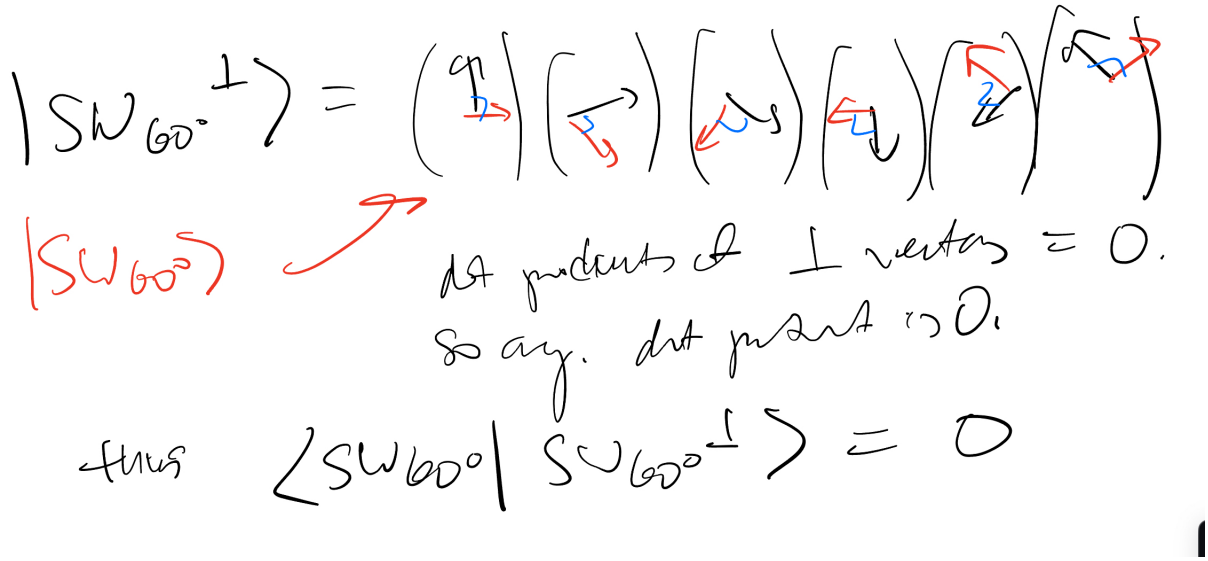
→ ↘ ↓ ↙ ← ↖ ↑ ↗

91 All the planes of rotation for IncrL

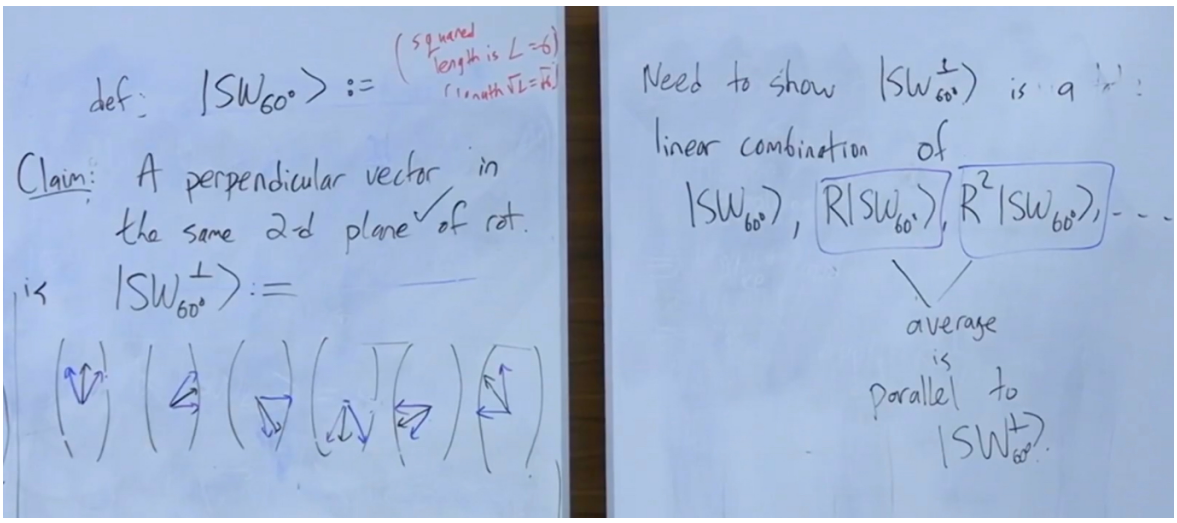
Video #091/100 Table of Contents

Now, let's find another vector that's in this same plane of rotation, but perpendicular so we can find a basis for this 2-D plane.

Claim: a perpendicular vector in the same 2-D plane of rotation is $\left|(SW_{60^\circ})^{\perp}\right\rangle$, which is the same as $|SW_{60^\circ}\rangle$ except you start at the vertical one.



We also need to check $\left|(SW_{60^\circ}^{\perp})\right\rangle$ is a linear combination of $R\,|SW_{60^\circ}\rangle, R^2\,|SW_{60^\circ}\rangle$.



Summary: we've found a perpendicular basis for one plane of rotation for R, by 60°. These are the two vectors $|SW_{60^\circ}\rangle, \left|(SW_{60^\circ}^{\perp})\right\rangle$. Generally, we can define the two $2L$-dimensional vectors $|SW_\theta\rangle$, $\left|(SW_\theta)^{\perp}\right\rangle$ for $\theta = \frac{1}{L}\tau$. Everything works the same so long as rotating by θ L times is a no-op. We can make a steering wheel for $\theta = 0, \frac{1}{L}\tau, \frac{2}{L}\tau, \ldots, \frac{L-1}{L}\tau$

Conclusion: for this $2L$-dimensional operator R, we've found L different 2-D planes of rotation with angles (as promised) $\frac{0}{L}\tau, \frac{1}{L}\tau, \frac{2}{L}\tau, \ldots, \frac{L-1}{L}\tau$. It remains to check that all these planes of rotation are perpendicular.

Recall that we're going to do Rotation Estimation on this R. But it has to be able to handle glitches, which is difficult because what if we don't know what plane to do the offset rotation in (Video #083/100)? But guess what! We know all the planes, since all it has to do is rotate the B qubit (Video #089/100)!

Next lecture, check the planes are perpendicular. Therefore, we should be able to know what happens when we do Rotation Estimation on R. However, we can't actually make any of these steering wheel states - it's going to be random for which one of the $\frac{i}{L}\tau$ states we start in. This randomness is uniformly distributed across possible values of i, because, as we'll show, $|start\rangle = |00001\rangle \otimes |0\rangle$ has an equal component in each plane, AKA the projection of $|start\rangle$ onto each of the 2-D rotation planes has equal length regardless of which rotation plane you project onto.

91.1 Exercises (continued from last one)

(c) Write out all 16 vectors $\left|SW_{\frac{K}{8}\tau}\right\rangle$ and $\left|SW^{\perp}_{\frac{K}{8}\tau}\right\rangle$ for $K = 0, 1, \ldots, 7$. Please display each of them in the conventional format of a column of numbers of height 16. You may display $\sqrt{\frac{1}{2}}$ as .7 if you like. Please list them consecutively in the order $|SW_0\rangle, \left|SW^{\perp}_0\right\rangle, \left|SW_{\frac{\tau}{8}}\right\rangle, \ldots, \left|SW^{\perp}_{\frac{7\tau}{8}}\right\rangle$. In fact, your final display should look like a 16×16 grid of numbers.

91.2 Solutions

	K	0		1		2		3		4		5		6		7	
state	Deg	0		-45		-90		-135		-180		-225		-270		-315	
0000		1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0001		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0010		1	0	0.7	0.7	0	1	-0.7	0.7	-1	0	-0.7	-0.7	0	-1	0.7	-0.7
0011		0	1	-0.7	0.7	-1	0	-0.7	-0.7	0	-1	0.7	-0.7	1	0	0.7	0.7
0100		1	0	0	1	-1	0	0	-1	1	0	0	1	-1	0	0	-1
0101		0	1	-1	0	0	-1	1	0	0	1	-1	0	0	-1	1	0
0110		1	0	-0.7	0.7	0	-1	0.7	0.7	-1	0	0.7	-0.7	0	1	-0.7	-0.7
0111		0	1	-0.7	-0.7	1	0	-0.7	0.7	0	-1	0.7	0.7	-1	0	0.7	-0.7
1000		1	0	-1	0	1	0	-1	0	1	0	-1	0	1	0	-1	0
1001		0	1	0	-1	0	1	0	-1	0	1	0	-1	0	1	0	-1
1010		1	0	-0.7	-0.7	0	1	0.7	-0.7	-1	0	0.7	0.7	0	-1	-0.7	0.7
1011		0	1	0.7	-0.7	-1	0	0.7	0.7	0	-1	-0.7	0.7	1	0	-0.7	-0.7
1100		1	0	0	-1	-1	0	0	1	1	0	0	-1	-1	0	0	1
1101		0	1	1	0	0	-1	-1	0	0	1	1	0	0	-1	-1	0
1110		1	0	0.7	-0.7	0	-1	-0.7	-0.7	-1	0	-0.7	0.7	0	1	0.7	0.7
1111		0	1	0.7	0.7	1	0	0.7	-0.7	0	-1	-0.7	-0.7	-1	0	-0.7	0.7

92 Recapping Factoring technical details (Lecture 24)

Video #092/100 Table of Contents

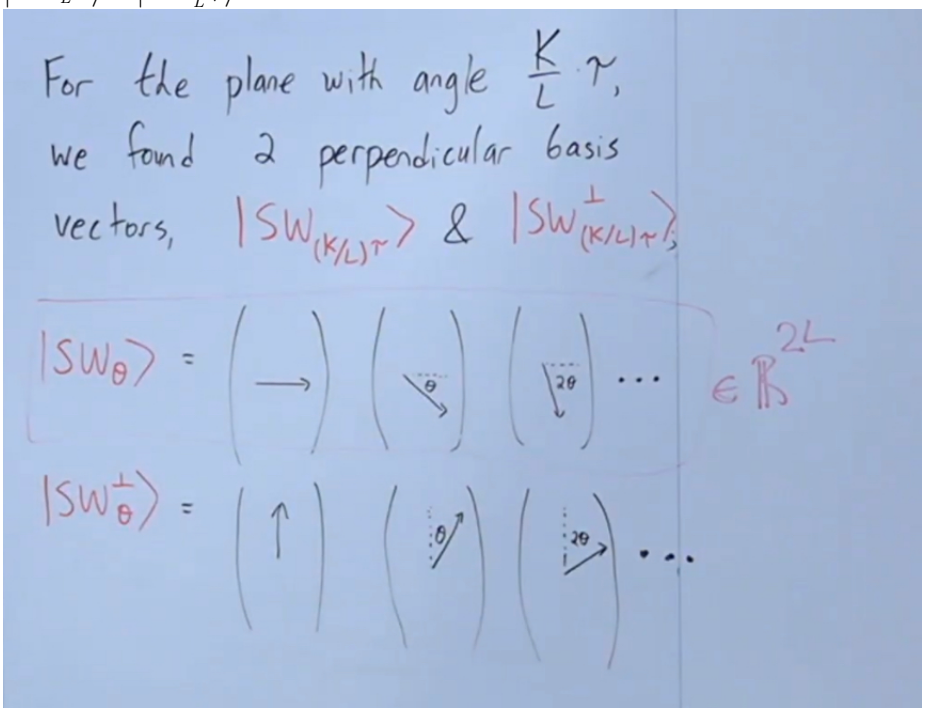
Plan for factoring $N = P \cdot Q$:

- Set $B = 2$
- Main goal: find first L such that $B^L \equiv 1 \bmod N$
- Then try $X = B^{\frac{L}{2}} \bmod N$. Usually, $\{P, Q\} = \gcd(X \pm 1, N)$, else try a few more random B's.
- Make quantum code for TimesBModN
- Let R be associated unitary operating on qubits $A_1, \ldots, A_n, A_{n+1}$.
- Claim: doing Rotation Estimation on R, $|start\rangle = |00\ldots01\rangle \otimes |0\rangle$, gives a "clue": many $(2n)$ digits for $\frac{K}{L}\tau$, for random K.
- A few "clues" are sufficient to determine L.

Recall: all the 'action' of R takes place in the $2L$-dimensional space spanned by basic states $|a_1 a_2 \ldots a_n\rangle \otimes |b\rangle$ where $a_1 a_2 \ldots a_n$ is a power of 2 (or B) mod N, $b \in \{0, 1\}$.

We discovered L planes of rotation for R, with angles $\left\{\frac{0}{L}\tau, \frac{1}{L}\tau, \frac{2}{L}\tau, \ldots, \frac{L-1}{L}\tau\right\}$. Remains to show that each of these planes or perpendicular.

For the plane with angle $\frac{K}{L}\tau$, we found 2 perpendicular basis vectors ("steering wheel" states) $\left|SW_{\frac{K}{L}\tau}\right\rangle$ & $\left|SW^{\perp}_{\frac{K}{L}\tau}\right\rangle$.



92.1 Exercises (continued from last one)

(d) Show full details of the calculation of the dot product $\left|SW_{(3/8)\tau}\right\rangle$ and $Incr8'\left|SW_{(3/8)\tau}\right\rangle$ and say a few words about what this means.

92.2 Solutions

		Incr8'	Dot product part 1	Dot product part 2	sum
state in base 10	SW 3/8 TAU	SW 3/8 TAU			
0	1	-0.7	-0.7		-0.7
1	0	-0.7		0	
2	-0.7	1	-0.7		-0.7
3	-0.7	0		0	
4	0	-0.7	0		-0.7
5	1	-0.7		-0.7	
6	0.7	0	0		-0.7
7	-0.7	1		-0.7	
8	-1	0.7	-0.7		-0.7
9	0	-0.7		0	
10	0.7	-1	-0.7		-0.7
11	0.7	0		0	
12	0	0.7	0		-0.7
13	-1	0.7		-0.7	
14	-0.7	0	0		-0.7
15	0.7	-1		-0.7	
				Average dot prod:	-0.7

The average of the 2-D dot products is equal to -0.7, which is a standin for $-\sqrt{\frac{1}{2}}$. The dot product of these two normalized states is the cosine of the angle between them. It follows that $-\sqrt{\frac{1}{2}} = \cos\theta$, which implies $\theta = 135°$. This makes sense because $(3/8)\tau$ is 135°.

93 IncrL's rotation planes: perpendicular

Video #093/100 Table of Contents

WTS: the planes of rotation that we found are perpendicular to each other. That is, if $\theta_1 = \frac{K_1}{L}\tau$ and $\theta_2 = \frac{K_2}{L}\tau$, where $\theta_1 \neq \theta_2$, then $|SW_{\theta_1}\rangle \perp |SW_{\theta_2}\rangle$, i.e. have dot product 0 $\langle SW_{\theta_1}|SW_{\theta_2}\rangle$.

Proof. The average of the 2-D vector dot products yields $\langle SW_{\theta_1}|SW_{\theta_2}\rangle$ (loosely dealing with unnoramlized states, but this isn't too big a deal). For tehse two vectors, the angle differences for each 2-D vector dot product are (letting $\Delta = \theta_2 - \theta_1$) $0, \Delta, 2\Delta, \ldots, (L-1)\Delta$. $\Delta = \frac{integer}{L}\tau$ for some integer which is not zero (since $\theta_1 \neq \theta_2$). Then the dot products being averaged are $\cos 0, \cos\Delta, \cos 2\Delta, \ldots, \cos(L-1)\Delta$.

Fact: this average is 0. Trick: draw a unit circle in $\mathbb{R}^2$. Consider L unit vectors whose angles are $0\Delta, 1\Delta, 2\Delta, \ldots, (L-1)\Delta$. Their tips are equally spaced around the circle. So their "center of mass" is the origin. This indicates the average of their x (and y, though this isn't useful) coordinates is zero, hence the average of their cosines are zero. So we're done! □

93.1 Exercises (continued from last one)

(e) Show full details of the calculation of the dot product $\left|SW_{(3/8)\tau}\right\rangle$ and $\left|SW^{\perp}_{(2/8)\tau}\right\rangle$ and say a few words about what this means.

93.2 Solutions

	part e		Dot product part 1	sum
state in base 10	SW 3/8 TAU	SW 2/8 TAU per	Dot product part 2	
0	1	0	0	0
1	0	1	0	
2	-0.7	1	-0.7	-0.7
3	-0.7	0	0	
4	0	0	0	-1
5	1	-1	-1	
6	0.7	-1	-0.7	-0.7
7	-0.7	0	0	
8	-1	0	0	0
9	0	1	0	
10	0.7	1	0.7	0.7
11	0.7	0	0	
12	0	0	0	1
13	-1	-1	1	
14	-0.7	-1	0.7	0.7
15	0.7	0	0	
			Average dot prod:	0

The average of the 2-D dot products is equal to zero, indicating that the two vectors are perpendicular. This makes sense, since they are in different 2-D planes of rotation for R. Woohoo!

94 Equal superposition of steering wheels

Video #094/100 Table of Contents

We don't know how to prepare qubits $A_1, \ldots, A_n, A_{n+1}$ in any $\left|SW_{\frac{K}{L}\tau}\right\rangle$. Sad. So we'll run Rotation Estimation on R, with qubits in state $|start\rangle = |00\ldots01\rangle \otimes |{\rightarrow}\rangle$ AKA $|00\ldots01\rangle \otimes |0\rangle$.

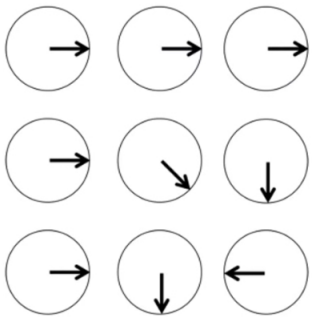
Claim: This $|start\rangle$ is an equal average of all the $|SW_\theta\rangle$ vectors, where $\theta = \frac{K}{L}\tau$.

Proof. $|start\rangle = \begin{pmatrix}\rightarrow\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \ldots, \begin{pmatrix}0\\0\end{pmatrix}$

Clearly, the average of $|SW_0\rangle$ first columns is $\begin{pmatrix}\rightarrow\end{pmatrix}$. For nonzero θ, all of the arrows cancel each other out! In the jth column, $j \neq 0$, we get the average of 2-D unit vectors at angle $j \cdot \frac{0}{L}\tau, j \cdot \frac{1}{L}\tau, j \cdot \frac{2}{L}\tau, \ldots, j \cdot \frac{L-1}{L}\tau = 0 \cdot \frac{j}{L}\tau . 1 \cdot \frac{j}{L}\tau, 2 \cdot \frac{j}{L}\tau, \ldots$. These are all equally spaced around the circle, center of mass/average is going to be $\begin{pmatrix}0\\0\end{pmatrix}$. So $|start\rangle$ (unnormalized) is $\frac{1}{L}\left|SW_{\frac{0}{L}\tau}\right\rangle + \frac{1}{L}\left|SW_{\frac{1}{L}\tau}\right\rangle + \frac{1}{L}\left|SW_{\frac{2}{L}\tau}\right\rangle + \ldots$ which are in planes $P_0, P_1, P_2, \ldots$ respectively where P_j is a 2-D plane of rotation angle $\frac{j}{L}\tau$. If we wanted normalized state, we would replace the $\frac{1}{L}$ with $\sqrt{\frac{1}{L}}$. So $|start\rangle$ has equal components on all the planes of rotation. □

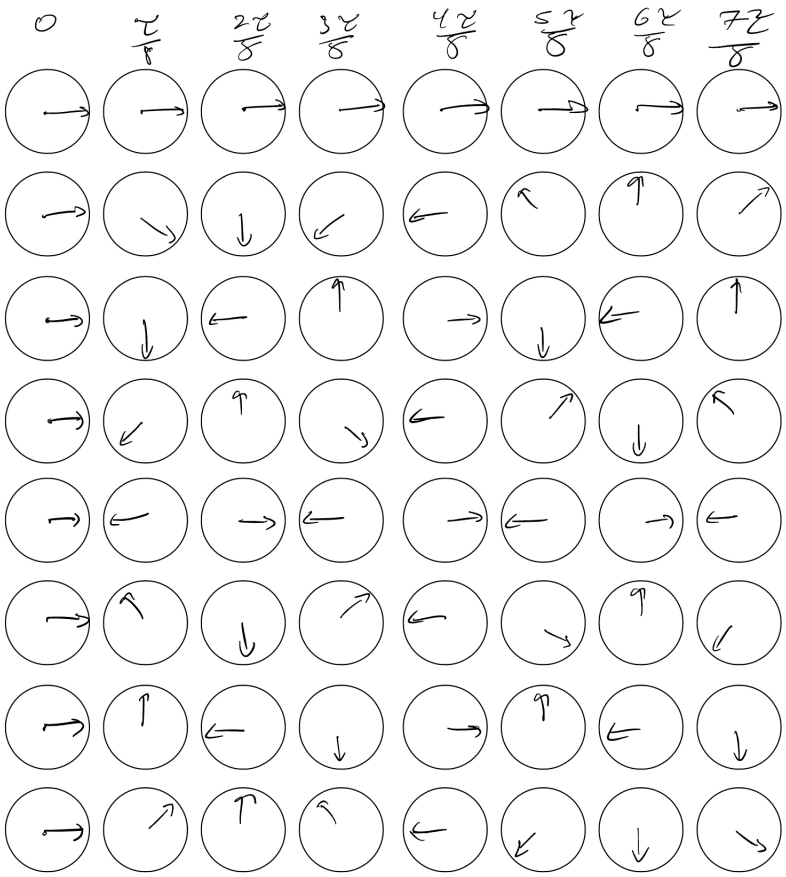
94.1 Exercises (continued from last one)

(f) Draw an 8×8 grid of "clocks", where the Kth column ($K = 0\ldots7$) depicts $|\mathrm{SW}_{(K/8)\tau}\rangle$ in the following way: Whereas in the lessons we depict it as a row of 8 two-d unit vectors, you should instead depict it as a column of 8 two-d unit vectors. Also, please draw a unit circle around each of the unit vectors. So the top-left part of your drawing should look like this...



I will use the convention that clockwise rotations are positive. This way I don't have to write a bunch of negative signs. This convention continues for the exercises in Video #095/100.

94.2 Solutions



95 Rotation Estimation in superposition: 1

Video #095/100 Table of Contents

Final Theorem: with this $|start\rangle$, Rotation Estimation outputs each of the rotation angles $\frac{K}{L}\tau$ with equal probability $\frac{1}{L}$.

Analogous Theorem (which is less notationally cumbersome): In this setting, if Rotation Estimation is performed on $|start\rangle$, each of $\theta_0, \theta_1, \theta_2, \theta_3$ is output with probability $\frac{1}{4}$.

for simplicity, let's imagine that $L = 4$ and the planes of rotation P_0, P_1, P_2, P_3 that R rotates are:

P_0 Span of $|000\rangle, |001\rangle$ with rotation angle θ_0

P_1 Span of $|010\rangle, |011\rangle$ with rotation angle θ_1

P_2 Span of $|100\rangle, |101\rangle$ with rotation angle θ_2

P_3 Span of $|110\rangle, |111\rangle$ with rotation angle θ_3

Now imagine $|start\rangle = \sqrt{\frac{1}{4}}\left(|000\rangle + |010\rangle + |100\rangle + |110\rangle\right)$.

Proof. of the analogous theorem: remember Rotation Estimation:

1. does "new qubit $D_1 \dots D_m$" for $m = O(n)$, "data qubits"
2. Does a unitary U on $A_1 \dots A_{m+1}$ (A_1, A_2, A_3), $D_1, D_2, \dots, D_m$
3. Extracts $D_1, D_2, \dots, D_m$
4. Classical alg. takes readouts and outputs $\hat{\theta}$.

Computing all answers in superposition, then extracting for a random one. □

95.1 Exercises (continued from last one)

The discrete Fourier transform. It's mathematically a bit unusual to talk about an 8×8 matrix of two-d unit vectors; usually you have *numbers* as the entries of a matrix. But there is a standard way to (try to) think of a two-d vector $\begin{bmatrix} x \\ y \end{bmatrix}$ as a number... as the *complex* number $x + iy$.

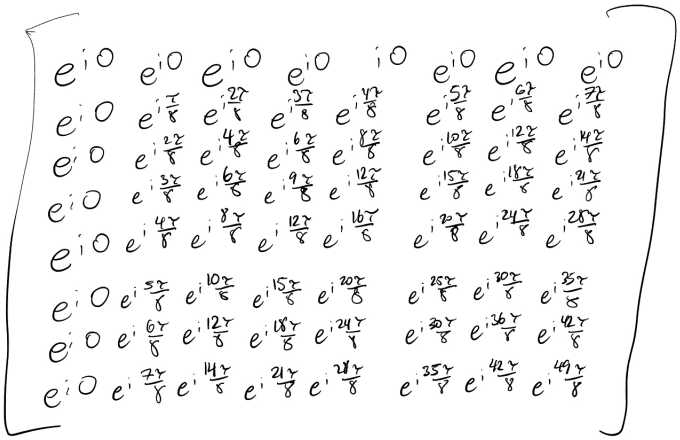
Actually, if it's a unit two-d vector, then you can write it as $\begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$ for some $0 \leqslant \theta < \tau$. In this case the associated complex number is $\cos\theta + i\sin\theta$, which you may know can also be written as $e^{i\theta}$.

(g) Write the 8×8 grid from (f) as an 8×8 matrix of complex numbers. Write each complex number as $e^{i\theta}$ for an appropriate value of θ. Remark: The resulting matrix is usually called DFT_8, the 8×8 Discrete Fourier Transform.

(h) Let $|\mathrm{SW}_K\rangle$ denote the Kth column of this matrix ($K = 0, 1, \dots, 7$), so $|\mathrm{SW}_K\rangle \in \mathbb{C}^8$. For each K, determine a complex number λ_K such that $\mathrm{Incr8} \cdot |\mathrm{SW}_K\rangle = \lambda_K \cdot |\mathrm{SW}_K\rangle$. Remark: thus the vector $|\mathrm{SW}_K\rangle$ is an eigenvector of Incr8, with eigenvalue λ_K.

95.2 Solutions

(g)



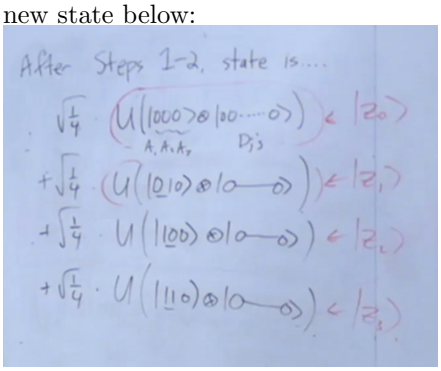
(h) For each K, $\lambda_K = e^{-iK\frac{\tau}{8}}$. Why? Incr8 shifts entries down (and brings the last entry to the top).

Then for each entry, we want to reassign it to the entry above it, which means we need to rotate in the opposite direction by $\frac{K\tau}{8}$. For example, when $K = 1$, we want the first entry to become $e^{-i\frac{1\tau}{8}}$, which is the same as $e^{i\frac{7\tau}{8}}$ because $\frac{7\tau}{8}$ and $-\frac{1\tau}{8}$ are coterminal angles.

96 Factoring finally finished!

Video #096/100 Table of Contents

Basically, Steps 1 and 2 do the algorithm separately on all 4 "pieces" of the starting state, resulting in this new state below:



- Fact 1: These $|z\rangle$'s are unit vectors that are perpendicular. We applied a unitary to a bunch of unit vectors.
- Fact 2: If the $D_1, \ldots, D_m$ qubits are $|z_j\rangle$ are extracted, the classical alg. processing results outputs θ_j (wowp).
- Fact 3: true for all $|z_j\rangle$. In, say, $|z_3\rangle$, all amplitude is on basic states starting with $|11\_**\cdots*\rangle$ where the 1s are the A_1, A_2 qubits, the A_3 qubit is $\_$ because it could be 0 or 1, and the stars are the D qubits. Think of how $|z_3\rangle$ comes to be. Initially, in the computation of $|z_3\rangle$ by doing $U(|110\rangle \otimes |0\ldots0\rangle)$, the claim holds. The big unitary U mostly does things that look like if D_j then R^{10^k} on A_1, A_2, A_3. Inductively, all the nodes you ever encounter will have states that start with 11.

  Inducitvely follows that in amplitude tree analysis, every node looks something like $|v\rangle \otimes |1010\ldots101011\rangle$. $|v\rangle$ is in $P_3 = span\,|110\rangle, |111\rangle$ so the state is gonna start with 2 ones.

So after steps 1-2, $|start\rangle$ has become $\sqrt{\frac{1}{4}}\left(|z_0\rangle + |z_1\rangle + |z_2\rangle + |z_3\rangle\right)$

Then what happens after Steps 3-4? Thought Experiment! After Steps 1-2, "Alice" gives $D_1 \ldots D_m$ to Dave the Data Analyst. Dave can do Steps 3-4 alone in his lab. Nothing Alice can do will affect the probability distribution of what Dave sees. Imagine Alice measures (non-destructively) just A_1, A_2. Alice will see: 00, 01, 10, or 11. Say she sees 11.

$\mathbb{P}(11)$ is the total squared amplitude on basic states starting with 11. By Fact 3, this is the total squared amplitude on $|z_3\rangle$. But Fact 1 tells us $|z_3\rangle$ is a unit vector, so the probability Alice sees 11 is $\sqrt{\frac{1}{4}}^2 = \frac{1}{4}$. And if Alice sees 11, the new state of $A_1 \ldots A_3, D_1 \ldots D_m$ is $|z_3\rangle$. When Dave does his thing, he'll output θ_3 (Fact 2).

So Dave's chance of seeing θ_i for i is 0, 1, 2, or 3 is going to be $\frac{1}{4}$ for all of them. Yay!

96.1 Exercises

(i) Show that the squared length of each $|\mathrm{SW}_K\rangle$, namely $\langle \mathrm{SW}_K|\mathrm{SW}_K\rangle$, is 8. Remember: the squared length/magnitude of a complex number z is $|z|^2 = z \cdot \overline{z}$, where $\overline{z}$ is the complex-conjugate of z. And remember that for complex vector $|\mathrm{SW}_K\rangle$, $\langle \mathrm{SW}_K|$ is not just the row-vector version of $|\mathrm{SW}_K\rangle$ but is the *complex-conjugated* row-vector version. (You might also recall that $\overline{e^{i\theta}} = e^{-i\theta}$.)

(j) Show that any two columns of DFT_8 are perpendicular; i.e., $\langle \mathrm{SW}_{K_1}|\mathrm{SW}_{K_2}\rangle = 0$ for $K_1 \neq K_2$. Remark: thus we have shown that the 8 vectors $\sqrt{\frac{1}{8}} \cdot |\mathrm{SW}_K\rangle$ form an orthonormal basis of $\mathbb{C}^8$.

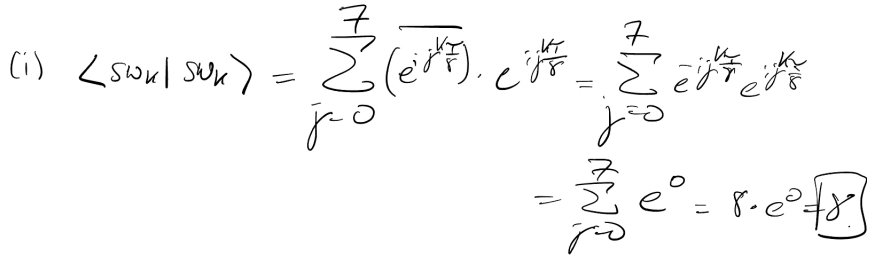
(k) Show that $U := \sqrt{\frac{1}{8}} \cdot \mathrm{DFT}_8$ satisfies $U^\dagger U = I$ (the identity matrix). Remark: the matrix $\frac{1}{8} \cdot U^\dagger$ is usually called the 8×8 Inverse Discrete Fourier Transform.

(l) In the Rotation Estimation part of the Quantum Factoring algorithm, we were really concerned about how the starting vector $|\mathrm{start}\rangle = (1,0,0,0,0,0,0,0)$ "broke up" into different components vis-a-vis $|\mathrm{SW}_0\rangle, |\mathrm{SW}_1\rangle, \ldots, |\mathrm{SW}_7\rangle$. Determining the components of a vector $|\mathrm{start}\rangle \in \mathbb{C}^8$ in the orthonormal basis $\{\sqrt{\frac{1}{8}} \cdot |\mathrm{SW}_K\rangle : K = 0, 1, \ldots, 7\}$ is literally the meaning of "taking the discrete Fourier transform" of $|\mathrm{start}\rangle$. Anyway, show that $|\mathrm{start}\rangle = (1,0,0,0,0,0,0,0)$ is equal to

$$\frac{1}{8} \cdot |\mathrm{SW}_0\rangle + \frac{1}{8} \cdot |\mathrm{SW}_1\rangle + \cdots + \frac{1}{8} \cdot |\mathrm{SW}_7\rangle.$$

96.2 Solutions

(i) See image below.



(j) See image above. Since the inner product is 0, the two vectors are perpendicular. This conclusion comes from Video #093/100, which showed that the sum (and the average) of all the $\cos j\Delta$ and $\sin j\Delta$ is zero.

(k) WTS $U^\dagger U = I$. Suffices to show $DFT_8^\dagger DFT_8 = 8I$. Using the fact that, $\langle SW_{K_1}|SW_{K_2}\rangle = 8$ iff $K_1 = K_2$ (part i) and 0 iff $K_1 \neq K_2$ (part j), we know that $DFT_8^\dagger DFT_8$ has all 8's along it's diagonal, and zeros everywhere else. That's just 8 times the 8×8 identity matrix, as desired.

(l) This is the same as just taking the row sums and putting it into the vector $|start\rangle$ But observe; in DFT_8, (which is unitary and also symmetric), the ith row is the same as the ith column. We've already deduced that the sum (and average) of all the complex numbers is zero if they form a sort of sequence with their angles (see part j, which cites Video 93). So we know all of the rows are 0, but wait - the first row is full of ones! So the first row actually sums to 8, because there is no sequence of angles (since the angle is 0)! So start, which is zeros everywhere except the zeroth entry, is just the sum of all the columns of DFT_8 (then divided by 8 to normalize).

97 Quantum algs recap: Hadamard Transform (Lecture 25)

Video #097/100 Table of Contents

Lecture 7: Given classical code operating on bits, you can convert to equivalent quantum code operating on qubits. Quantum computers are at least as efficient as classical computers.

Lecture 8: Easy to produce the uniform superposition over all 2^n n-bit strings. Then you can get superposition over all 2^n input-output pairs for your code. Computing in superposition is a great benefit over classical and even probabilistic computing.

Amplitudes can be negative, so subsequent manipulations and have some cancel out and some be amplified.

Lectures 6-9: The Hadamard Transform paradigm. Given code computing $f : \{0,1\}^n \to \{0,1\}$,

- prepare the uniform superposition
- if f then minus

  Now the 2^n amplitudes are ± 1 truth table values of f
- Hadamard all, then extract all
- Resulting amplitude on all zeros string (before extract) was the bias of f, and more generally, for the amplitude on bitstring $|b_1 b_2 \dots b_n\rangle$, it is equal to the correlation between f and $XOR_{b_1 b_2 \dots b_n}$. Bitstrings whose bitmasked XOR function that have high correlation or anticorrelation with f have higher probability of being output.

Uses of Hadamard Transform:

1. Bias-busting (Deutsch-Jozsa): is the bias of the code nonzero or zero? Given code for f, outputs "Biased!" or "No Comment".

   No false positives, and has positive power: if f's truth table is biased, the probability it outputs "Biased!" is greater than 0.

   Why is bias busting important? (HW 4.6) SAT le_p Bias-Busting - solving bias-busting efficiently means being able to solve SAT efficiently.

   Major speedup over classical algorithms though? No. It's possible that the probability of outputting "Biased!" could be positive, but really small. Hence why it's not practically interesting. But Bias-Busting is theoretically interesting. (Ogihara-Toda 1990) Assuming $P^{NP} \neq NP^{NP}$, no classical algorithm can have this same (useless) behavior.

   So there's something quantum code can do that classical code can't. Lovely!

2. Lectures 1,9: Mystery Toggles (Bernstein-Vazirani): Given code for f computing some bitmasked $XOR_{b_1 b_2 \dots b_n}$, a quanutm algorithm can learn the bitmask $b_1 \dots b_n$ with just one call to the code.

   Again, not an important task. It's an invented problem.

   Major speedup over classical algorithms though? Yes and no. A classical algorithm would have to make n calls to the code, and the quantum algorithm only does 1 call. However, the quantum algorithm still has $O(n)$ instructions. So if the MysteryToggles takes T steps to run, then quantum is $O(n+T)$ and classical is $O(nT)$. It's a nice enough speedup.

   This is theoretically interesting, showing a large gap when we count number of times to run mystery code (the "query model").

3. Simon's Algorithm (not taught in this class): similar intentionally designed function like MysteryToggles, where quantum needs only n calls while classical needs about 2^n calls.

TLDR: the first half of the course was to have quantum stuff interfere and cancel in nice ways.

97.1 Question

What do you think? If your friend asked you what made quantum computing powerful, would you tell them about the Hadamard Transform?

98 Quantum algs recap: Grover's alg and SAT

Video #098/100 Table of Contents

The second part of the course is more geometry. Even Bias-Busting is already geometric.

Recall Bias-busting: $|f\rangle = \sqrt{\frac{1}{2^n}}(+1,-1,-1,-1,+1,\dots)$ and $|unif\rangle = \sqrt{\frac{1}{2^n}}(+1,+1,+1,+1,+1,\dots)$. If f is un-biased, the dot product with $|unif\rangle$ is 0, so the 2 vectors are at angle 90°. Otherwise, the angle between them isn't 90 degrees.

Lecture 13: Elitzur-Naidman Bomb. If you rotate $|0\rangle$ by angle of ε the measure, the probability of seeing 1 is $\sin^2\varepsilon \approx \varepsilon^2$. But if you rotate $\frac{\frac{\pi}{2}}{\varepsilon}$ times, and then measure, you have a high probability of seeing 1. Then you can detect the bomb with high confidence by sort of "spreading out" your risk of setting off the bomb. We also showed $\frac{1}{\varepsilon}$ "work" can overcome ε^2 probabilities. This square root improvement is a hallmark of quantum speedups.

This is a quantum algorithm for a quantum problem (moving away from solving classical problems using quantum computation).

Lecture 14: Grover's Algorithm for SAT. Given classical code C computing a boolean function that takes in n input bits and outputs 1 bit, can distinguish $p = 0$ fraction of satisfying strings or $p \geq \frac{1}{2^n}$ fraction of satisfying strings, with work $\approx \sqrt{2^n} = (1.41\dots)^n$.

This is an important task, and is kind of a major speedup (still exponential time), but is of great theoretical itnerest since SAT is such a well-studied problem.

Conjectures of the form "Classical algorithms can't solve SAT in [blank] time"

- anything like n^{1000} (this is the crux of $P \neq NP$)
- anything like $(2^n)^{.999}$ (this is the "ETH", exponential time hypothesis)
- anything like $(1.999)^n$ (this is "SETH", strong ETH)

Grover's shows that quantum algorithms can break SETH. There's a result that preceded Grover's, that showed that any quantum algorithm that uses the input code C as a black box requires as least $\sqrt{2^n}$ time. You need to be able to look at the code to beat $\sqrt{2^n}$.

QSETH (quantum strong ETH) in 2019: no quantum algorhtm for SAT in time like $(\sqrt{2} - 0.0000001)^n$.

98.1 Question

Sometimes I think that "quantum computing" could just be called "rotational computing", or maybe "geometric computing". What do you think?

99 Quantum algs recap: Factoring

Video #099/100 Table of Contents

Lectures 15-17, 21: Rotation Estimation. Powerful paradigm. Can make an ultimate form of Grover: given code C with a p fraction of satisfying strings, can estimate p to $\pm\varepsilon$ with running C in $O(\frac{\sqrt{p}}{\varepsilon}$ times.) Chebyshev's inequality will show that classical algorithms required $O(\frac{p}{\varepsilon^2})$ (another square root speedup!). Might be a while though; classical computers are so developed and have such fast clock speeds that this speedup may not be practical for awhile.

If you happen to know a very efficient algorithm to repeatedly do R^{10^n}, you can make Rotation Estimation efficiently estimate rotation angles to n digits of precision ($\pm\varepsilon = \pm\frac{1}{10^n}$).

Lectures 22-24: Quantum algorithm for factoring n-digit numbers that is in $O(n^3)$ time. (Shor's uses Quantum Fourier transform, Kitaev's uses rotation, but in the exercises for Video #095/100, we see that the rotation stuff leads to the DFT). This is an important task, especially in cryptography, and is definitely a major speedup! The theoretical interest is that $P \subseteq NP$ (NP contains problems with efficient way to verify solutions, P contains problems with efficient ways to find solutions). We want to find NP problems that might not be in P (AKA NP-complete). We know that if one NP-complete problem is actually in P, then all NP-complete problems are in P.

There's one problem not known to be in P, but not known to be NP-complete. That's factoring.

Computing in superposition is useful in factoring because we can randomly create vectors in the planes of rotation, but they're random. Still better than probablistic computing though, since we don't even know what we'd try to create there.

99.1 Question

Why do you think quantum computers can factor efficiently? What would you say the "real reason" is?

100 Why study quantum computing?

Video #100/100 Table of Contents

Do we know any more quantum algorithms for classical problems? Nope. There's so many uses for quantum algorithms, but they all have downsides:

- Quite modest speedups, like a $(1.7)^n$ algorithm for the traveling salesperson problem (something about some quantum version of RAM)
- Problem that was extremely obscure, not clear what the algorithm is good for
- Exponential speedup for important problems, but there's fine print. E.g., HHL algorithm for solving systems of linear equations. People who believe in quantum machine learning point out the exponential speedup for this problem, but you don't really get a solution, you have to have a special form of matrix, you only get to sample from the solution, and basically, once you overcome the fine print, you reach a classical algorithm that does just as well.

  Additionally, the algorithm is heuristic, which is fine in many fields (SAT and ML often use heuristic algorithms). When quantum computers come, we'll see what happens!
- or maybe these algorithms don't exist yet!

Then why study quantum computing? Well there's not too many quantum algorithms for classical problems, but what about quantum algorithms for quantum computing? This transcends classical computing and P and NP - we're not even working on regular bitstrings anymore. Also, the universe is governed by quantum physics, so good stuff to know about quantum computing.

What's really in between solving NP-complete time and doing nothing? What's there?