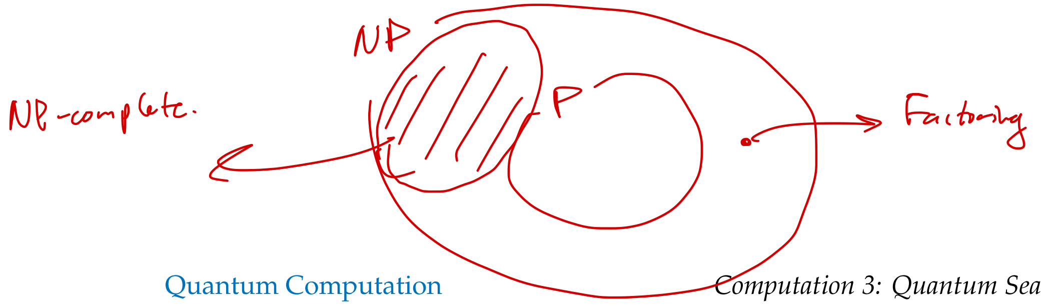
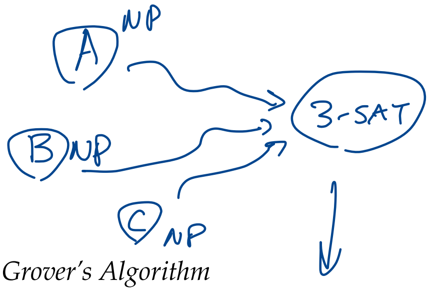




Efficient alg. for 3-SAT would be efficient alg. for ALL NP problems.

❊ Computation 3: Quantum Search - Grover's Algorithm

Learning Outcomes
Upon following these notes and the corresponding lecture, students will be able to

- model NP complete problems as search problems.
- describe how to translate this search problem into the query model.
- describe geometrically how a quantum algorithm can accomplish search in $O(2^{n/2})$ queries.
- construct circuits to implement Grover's algorithm.
- describe the implications of generalizing the search problem for Grover's algorithm.

3.1 The Search Problem

Once Shor's algorithm was discovered, many researchers became very interested in the potential of quantum computers to solve problems that are believed to be hard for classical computers to solved. Formally, the question being asked was the following: "Can quantum computers solve NP-complete problems?"

Example 3.1. 3-SAT

False.

- **Input:** A Boolean formula ϕ in 3-CNF form consisting of n-Boolean variables: $\phi(x_1, x_2, \ldots, x_n)$.

$$\text{3-CNF: } (x_2 \vee \overline{x}_7 \vee x_3) \wedge (\overset{0}{x_5} \vee \overset{0}{x_7} \vee \overset{0}{x_9}) \wedge \cdots \tag{91}$$

- **Output:** Does ϕ have a satisfying assignment? That is, is there a way of setting each x_i such that ϕ evaluates to TRUE?

I can "prove" to you that there is a satisfying assignment by giving you an assignment of the variables $x_1, \ldots, x_n$, which you just need to plug into the function and check for yourself.

The above example is important because not only is it in NP, it is actually a problem in a subset of NP problems called NP-complete. NP-complete problems are often referred to as "the hardest problems in NP", and is a significantly important set of problems because we know that an efficient solution to *any* NP-complete problem would mean there is an efficient solution to *every* problem in NP! It is generally believed that it is impossible to solve NP-complete problems in polynomial time using a classical computer.

The final algorithm we study this class is again a query based algorithm. Per usual, we will have black-box access to the function $f : \{0,1\}^n \to \{0,1\}$ we are interested in in the form of a unitary U_f:

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle . \tag{92}$$

We are interested in deciding some property of the function. For this last setting, we are interested in the search problem:

Definition 3.2 (Search Problem)**.** Let $f : \{0,1\}^n \to \{0,1\}$. Is there an n bit string x such that $f(x) = 1$?

This problem is interesting because any decision problem (including NP-complete ones) can be formulated in this way. For example, if we have a 3-SAT instance we can set $f(x_1 \cdots x_n) = \phi(x_1, \cdots, x_n)$. To simplify the analysis, we will first assume that f either has a unique solution or no solution at all. In other words,

$$|\{x|f(x) = 1\}| = 0 \text{ or } 1. \tag{93}$$

In the case that a solution does exist, we will denote it by the variable a.

Question 51. What is the classical query complexity for the above problem?

2^n queries.

You may be wondering if this is *too* general for analyzing NP-complete problems, but there is a widely believed hypothesis that a brute force search is the best we can do for 3SAT. In other words, we believe that the lower bound shown above for general f holds for functions that are NP-complete too. We now want to ask what kind of speed-up is possible if we use a quantum computer in this setting.

3.2 Grover's Algorithm

Let $|a\rangle$ be the standard basis state representing the value such that $f(a) = 1$. We will also use the notation

$$\rightarrow \quad |\psi\rangle := \frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} |x\rangle \qquad (94)$$

to represent the uniform superposition state.

One key fact about this algorithm is that the state of our system will **always** stay in the space spanned by $|a\rangle$ and $|\psi\rangle$.

→ **Question 52.** Letting $|v\rangle$ be the state of our system at any given time step, how can we express the above fact mathematically?

$$|v\rangle = \alpha|a\rangle + \beta|\psi\rangle$$

$$|x\rangle = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \rightarrow \text{position } x,$$

$$|01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Note that $|a\rangle$ and $|\psi\rangle$ are not orthogonal.

→ **Question 53.** What is the overlap between $|a\rangle$ and $|\psi\rangle$? Express this using trigonometric functions. Draw the two vectors, with $|a\rangle$ as a vector pointing upward, and $|\psi\rangle$ in the right direction. What does this say about the **probability** of measuring $|a\rangle$?

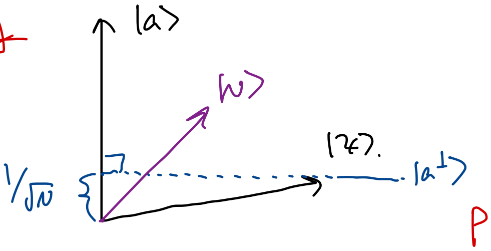
Overlap

$$\langle a|\psi\rangle = \langle a|\left(\frac{1}{\sqrt{N}}\sum_x |x\rangle\right)$$

$$= \frac{1}{\sqrt{N}}\sum_x \langle a|x\rangle$$

$$= \frac{1}{\sqrt{N}}\left(1 + \sum_{x\neq a} 0\right) = \frac{1}{\sqrt{N}}$$

Drawing



$$\left(1/\sqrt{N}\right)^2$$

Probability. $= 1/N = 1/2^n$

We can create an orthonormal basis that represents the same subspace by subtracting the $|a\rangle$ component from $|\psi\rangle$.

Question 54. Create a new state $|e\rangle$ that is orthonormal to $|a\rangle$ by subtracting the $|a\rangle$ component from $|\psi\rangle$.

$$|\hat{e}\rangle = |\psi\rangle - \langle a|\psi\rangle|a\rangle$$

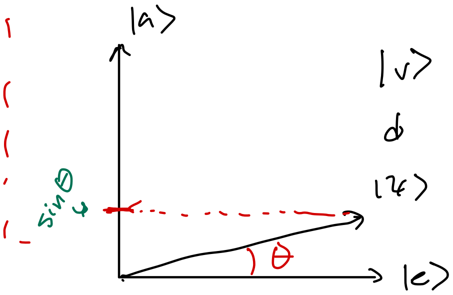
$$= \frac{1}{\sqrt{N}}\sum_x |x\rangle - \frac{1}{\sqrt{N}}|a\rangle$$

$$= \frac{1}{\sqrt{N}}\sum_{x\neq a} |x\rangle$$

Normalize!

Need equal probability of measuring all N-1 strings.

$$|e\rangle = \frac{1}{\sqrt{N-1}}\sum_{x\neq a} |x\rangle$$



Practice!

$$|\psi\rangle = \_\,|a\rangle + \_\,|e\rangle$$

The last preparation step we need is to express the angle between $|e\rangle$ and $|\psi\rangle$. For very small values of θ, it is known that

$$\sin\theta \approx \theta. \tag{95}$$

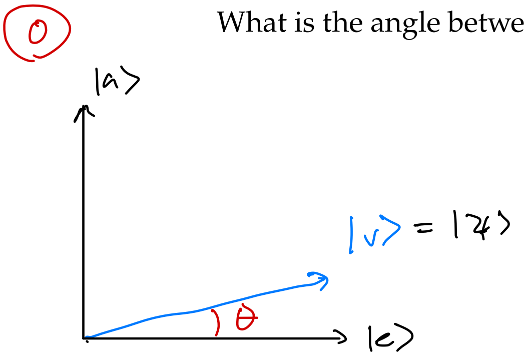
~~From this, we can conclude that the angle formed between $|\psi\rangle$ and $|e\rangle$ is approximately θ.~~

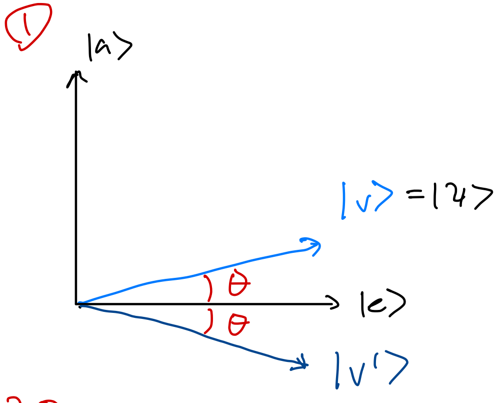
The main steps of the algorithm can be described purely geometrically. We will discuss how to implement these steps later, but for now let's build the intuition behind what the algorithm is doing. I will use $|v\rangle$ to represent the current state of the system. Here are the steps that we repeat:

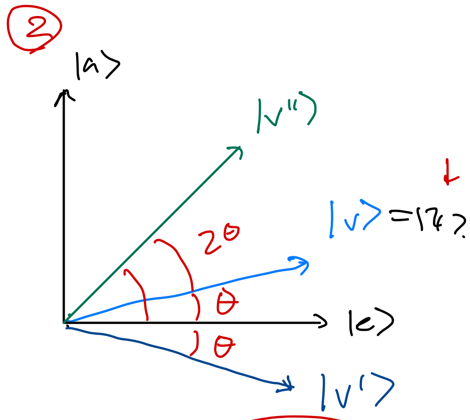
For ...

1. Reflect $|v\rangle$ over the $|e\rangle$ axis. to get $|v'\rangle$.
2. Reflect $|v'\rangle$ over $|\psi\rangle$.

Question 55. Let θ be the angle between $|v\rangle$ and $|e\rangle$ before applying the two steps above. What is the angle between the two *after* we apply the two steps?



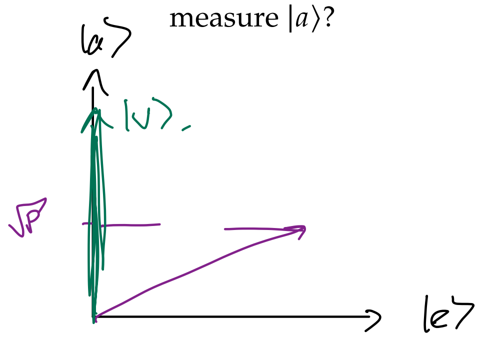




$3\theta = \theta + 2\theta$

After k repetitions, $\boxed{\theta + k2\theta}$

Question 56. What angle between $|v\rangle$ and $|e\rangle$ would give us the highest probability to measure $|a\rangle$?



Goal is $\pi/2$ radians between $|v\rangle$ and $|e\rangle$.

$\theta = 1/\sqrt{N}$.

$\theta + 2k\theta = \pi/2$

If we know θ, we can solve for # of iterations k.

Assuming we have $|v\rangle$ with $\overset{\theta}{=}\pi/2$.

$|a\rangle$ with probability 1.

not $|a\rangle$ " " 0.

3.2.1 Implementing Reflections

How can we implement the reflections? Before discussing that, let's think about how to express the action of these reflections mathematically. Suppose we have a state $|v\rangle$ that we want to reflect over some other state $|\phi\rangle$.

Question 57. Write down $|v\rangle$ as a superposition of $|\phi\rangle$ and $|\phi^{\perp}\rangle$. Write down the state $|v'\rangle$ after the reflection is completed.

3.2.2 Reflection over $|e\rangle$

The first reflection we need to do is over the $|e\rangle$ axis. That is, we want

$$|v\rangle = \alpha\,|e\rangle + \beta\,|a\rangle \tag{96}$$

$$|v'\rangle = \alpha\,|e\rangle - \beta\,|a\rangle \tag{97}$$

$$\tag{98}$$

In words, we want to multiply the $|a\rangle$ state by -1, and the rest of the states by +1. Remember that a is the unique bitstring satisfying $f(a) = 1$. We can summarize this action by

$$|v\rangle = \sum_x \alpha_x\,|x\rangle \longrightarrow |v'\rangle = \sum_x \alpha_x(-1)^{f(x)}\,|x\rangle\,. \tag{99}$$

Question 58. We know we can perform the above operation with access to U_f. U_f acts on $n+1$ qubits. What state does this qubit need to be in for us to apply the phase of $(-1)^{f(x)}$?

3.2.3 Reflection over $|\psi\rangle$

The mapping we want to perform here is

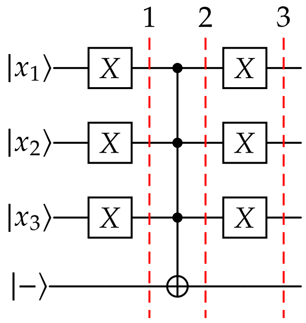
$$|v\rangle = \alpha|\psi\rangle + \beta|\psi^\perp\rangle \tag{100}$$
$$|v'\rangle = \alpha|\psi\rangle - \beta|\psi^\perp\rangle \tag{101}$$
$$\tag{102}$$

where $|\psi\rangle = \frac{1}{\sqrt{N}}\sum_x |x\rangle$.

Since $|\psi\rangle$ is generated by applying n Hadamard gates, the following is true:

$$H^{\otimes n}|0\cdots 0\rangle = |\psi\rangle \iff |0\cdots 0\rangle = H^{\otimes n}|\psi\rangle. \tag{103}$$

If we apply n Hadamard gates to $|\psi^\perp\rangle$, we get a state $|\phi\rangle$ which is a uniform superposition over all states perpendicular to $|0\cdots 0\rangle$. The goal is to now apply a phase of -1 to every standard basis state which is not equal to $|0\cdots 0\rangle$. I claim that the following 3 qubit circuit accomplishes this:



If input state is $|000\rangle$:

If input state is not $|000\rangle$:

3.3 Generalized Search

What about the (more likely) case where there are multiple solutions? That is,

$$|\{x|f(x) = 1\}| = M > 1. \tag{104}$$

Assume (for now) that the number of solutions M is known. We would like to find any one of these solutions. We again define two orthogonal states, the uniform superpositions of the solution states and non-solution states respectively:

$$|\phi_1\rangle = \frac{1}{\sqrt{M}} \sum_{x:f(x)=1} |x\rangle \quad |\phi_0\rangle = \frac{1}{\sqrt{N-M}} \sum_{x:f(x)=0} |x\rangle . \tag{105}$$

Then our starting uniform superposition state can be expressed as a weighted sum of the above two states as

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}\, |\phi_0\rangle + \sqrt{\frac{M}{N}}\, |\phi_1\rangle . \tag{106}$$

Since we know M, we can select the number of iterations c such that

$$(2c+1)\theta \approx \frac{\pi}{2} \tag{107}$$

in the same way we did in the previous section. In the case where $M << N$, $\theta \approx \sqrt{\frac{M}{N}}$, so the total number of iterations required is $O\left(\sqrt{\frac{M}{N}}\right)$.

Note: Since each iteration of Grover's algorithm advances the current state by 2θ, we can only guarantee that the final state is within θ of $|\phi_1\rangle$.

In the next section, we will look at an algorithm which will let us handle the case where M is unknown.

Before that though, we must resolve the question we started the section with. From the results in this section, Grover's algorithm does not provide us with an exponential speed-up over classical algorithms for search problems in the query model. Is there a quantum algorithm which can outperform Grover? It turns out this is not possible, and Grover's algorithm is "the best you can do" in this query model. There is a lower bound (which we will not prove here) that states that even if you are guaranteed that the $|s_f|$ (the number of satisfying solutions) is 0 or 1, there is not much we can do to improve.

More formally, the lower bound states that any quantum circuit that can determine between $|s_f| = 0$ and $|s_f| = 1$ with high probability will require $\Omega\left(\sqrt{N}\right)$ queries to U_f.

3.4 Amplitude Estimation

In the previous section, we mentioned that if we want to *find* an x such that $f(x) = 1$, we need to have an approximate idea of what M is. There is a standard method for accomplish this that uses the QFT, but instead of discussing that I will outline a result that was published by myself and Sandy, along with two (then) undergrads who took this class two years ago.

Our algorithm is **iterative**, meaning that it jumps between a parameterized quantum portion and a classical postprocessing portion. Let a be the solutions, and θ_a be the angle formed between the starting state $|\psi\rangle$ and the superposition $|\phi_0\rangle$ over states that evaluate to 0. The two are related by the expression

$$a = \sin\theta_a. \tag{108}$$

If we know a, we know how many queries to make in the generalized search algorithm. The algorithm will output an estimate for θ_a. We do this by maintaining lower and upperbounds for what θ_a is.

Picture of how the algorithm works: