CSCI 4140
Quantum Algorithms
Part Two

Mark Green

Faculty of Science

Ontario Tech

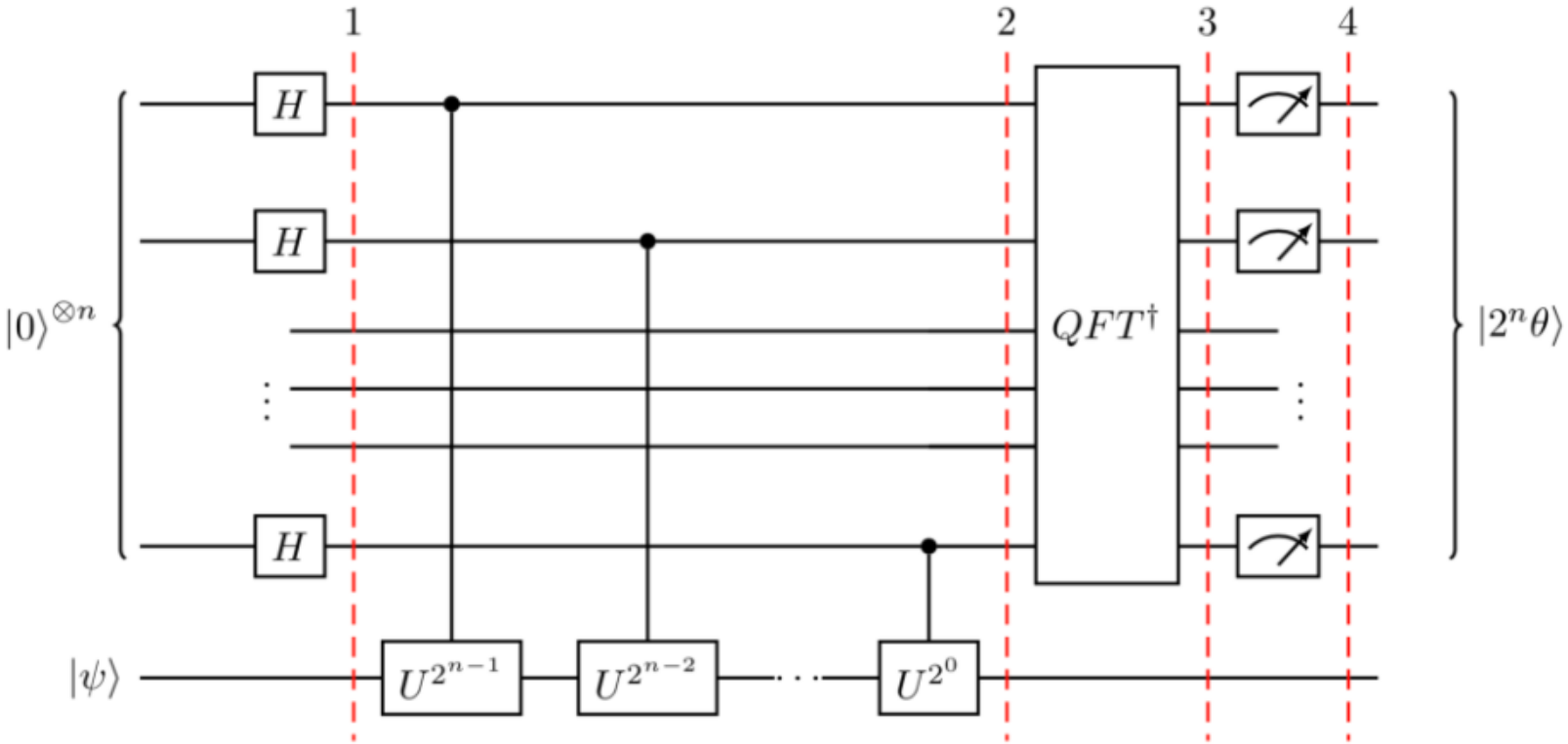
Introduction

- In this part of the course we will examine two of the most important quantum algorithms; Shor's algorithm and Grover's algorithm
- Shor's algorithm is the one that is causing all the problems for the security community
- But first, we will examine quantum phase estimation, which is an important part of Shor's algorithm

Quantum Phase Estimation

- This is a key part of many quantum algorithms, a useful subroutine which builds on the quantum Fourier transform
- Given and unitary operator U which has an eigenvector $|\psi\rangle$ quantum phase estimation approximates θ in $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$
- Since we have a limited number of bits, we may not be able to obtain the exact value of θ, remember we can only measure 0 and 1
- As a result the algorithm's result is $|2^n\theta\rangle$ where n is the number of qubits
- The circuit is shown on the next slide

Quantum Phase Estimation



Quantum Phase Estimation

- Note the use of the inverse quantum Fourier transform
- So how does this work?
- There are two inputs to this algorithm
  - An n bit register we call the counting register
  - The eigenvector $|\psi\rangle$
- It uses a sequence of controlled U gates and phase kickback to write the phase of U, in the Fourier basis, into the counting register
- The inverse Fourier transform is then used to convert this to the phase

Quantum Phase Estimation

- Recall that in the QFT on n bits, the first bit does a full rotation when going from 0 to 2^n, the next bit does two full rotations, the third bit does 4 full rotations, etc.
- If we have a value x between 0 and 2^n the first qubit will rotate by $\frac{x}{2^n}$, the next bit by $\frac{2x}{2^n}$, the third bit by $\frac{4x}{2^n}$, etc.
- When we use the controlled U gate and kickback the rotation will be proportional to the phase $2^{2\pi i\theta}$ and its multiples
- All we need to do is apply the inverse QFT to obtain the phase

Quantum Phase Estimation

- That's the high level view of what's going on, now let's turn to the math behind it
- Out circuit starts out in the following state:

$$\psi_0 = |0\rangle^{\otimes n}|\psi\rangle$$

- Next we apply the Hadamard gates to get:

$$\psi_1 = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\otimes n}|\psi\rangle$$

Quantum Phase Estimation

- Since U is a unitary operator and $|\psi\rangle$ is one of its eigenvectors we have the following:

$$U^j|\psi\rangle = U^{j-1}|\psi\rangle = U^{j-1}e^{2\pi i\theta}|\psi\rangle = \ \ldots = \ e^{2\pi i\theta}|\psi\rangle$$

- Next we perform n controlled gates $C - U^{2^j}$
- make the following observation:

$$|0\rangle \otimes |\psi\rangle + |1\rangle \otimes e^{2\pi i\theta}|\psi\rangle = (|0\rangle + e^{2\pi i\theta}|1\rangle) \otimes |\psi\rangle$$

Quantum Phase Estimation

- The result of applying the controlled gates is:

$$\psi_2 = \frac{1}{\sqrt{2^n}}\left(|0\rangle + 2^{2\pi i\theta 2^{n-1}}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + 2^{2\pi i\theta 2^1}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i\theta 2^0}|1\rangle\right) \otimes |\psi\rangle$$

- Or:

$$\psi_2 = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i\theta k}|k\rangle \otimes |\psi\rangle$$

- Note that this is the same as $QFT|x\rangle$ where $x = 2^n\theta$

Quantum Phase Estimation

- After applying the inverse QFT we have the following:

$$\psi_3 = \frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{-\frac{2\pi i k}{2^n}(x-2^n\theta)} \otimes |\psi\rangle$$

- The above expression peaks when the exponent is zero, which is when $x = 2^n\theta$
- If $2^n\theta$ is an integer we will measure the following with high probability:

$$\psi_4 = |2^n\theta\rangle \otimes |\psi\rangle$$

Quantum Phase Estimation

- If $2^n\theta$ is not an integer we will still get a peak close to the phase with a probability of $\frac{1}{\pi^2}$
- With this in hand we can now move on to Shor's algorithm

Shor's Algorithm

- This is THE algorithm, the one that has caused all of the excitement and worry about quantum computing
- Modern cryptography algorithms are based on hard to solve mathematical problems
- If it takes decades or centuries for a computer to solve the problem the code is considered to be secure
- Even if the messages are intercepted, the time required to decode them exceeds the usefulness of the data

Shor’s Algorithm

- All of our commercial interactions depends on these codes, they are very important
- The the popular codes are based on factoring large numbers, the key is to find the prime factors for these numbers
- This is a computationally difficult problem, so these codes have been viewed as secure
- Shor’s algorithm can perform this factorization on a suitable quantum computer in a reasonable length of time
- This essentially breaks the existing codes

Shor’s Algorithm

- While Shor’s algorithm has been around since 1997 it has only attracted a lot of interest recently
- When it was first published there were no quantum computers, it’s only been in the last few years that they have been beginning to show promise
- It has been estimated that a quantum computer large enough to run Shor’s algorithm on real codes will be constructed in the next decade or so
- This is now a real threat to security

Shor's Algorithm

- Shor's algorithm is interesting for the following two reasons:
  - It requires the cooperation of a quantum and a classical computer, the algorithm runs on both computers
  - It is a probabilistic algorithm in the sense that the quantum part may need to run several times before it comes up with a non-trivial solution
- The algorithm is shown on the next slide
- The input to this algorithm is the number N to be factored

Shor's Algorithm

1. If N is even, return 2
2. If $N = p^q$, return p. This can be done in polynomial time on a classical computer
3. Choose a random number a such that $1 < a \leq N - 1$, if gcd(a,N)>1 return gcd(a,N). This can be done on a classical computer
4. Use the order finding quantum algorithm to find the order r of a modulo N
5. If r is odd or $a^{\frac{r}{2}} = -1(mod\ N)$ go back to step 3. Otherwise compute $\gcd\left(a^{\frac{r}{2}} + 1, N\right) and\ \gcd(a^{\frac{r}{2}} - 1, N)$, if either is a non-trivial factor of N, return it. Otherwise back to step 3

Shor's Algorithm

- The quantum part of the algorithm is in step 4, the order or period finding problem, two names for the same problem
- Consider the following function:

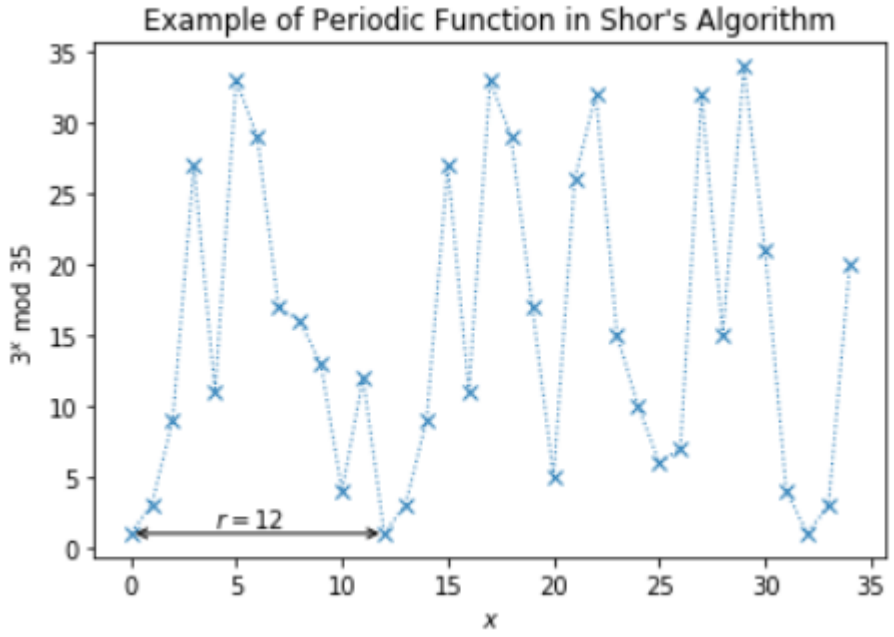
$$f(x) = a^x mod\ N$$

- Where:
  - $a < N$
  - a and N have no common factors
- The period of f(x) is the smallest non-zero integer r, such that

$$a^r mod\ N = 1$$

Shor’s Algorithm

- The following graph shows a plot of f(x) with a=3 and N=35



- In this case we can see that the period is 12

Shor's Algorithm

- Shor's algorithm is based on using quantum phase estimation on the following unitary operatory:

$$U|y\rangle = |ay \; mod \; N\rangle$$

- For our example function this operates in the following way:

$$U|1\rangle = |3\rangle$$
$$U^2|1\rangle = |9\rangle$$
$$\dots$$
$$U^r|1\rangle = |1\rangle$$

Shor's Algorithm

- The superposition of all the states in this cycle would be an eigenvector of U, this gives:

$$|u_0\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |a^k \ mod \ N\rangle$$

- Unfortunately, the corresponding eigenvalue is 1, which isn't very helpful
- A more interesting eigenstate has a different phase for each of the computation basis states:

$$|u_1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{2\pi i k}{r}} |a^k mod \ N\rangle$$

Shor's Algorithm

- Now we have the following:

$$U|u_1\rangle = e^{\frac{2\pi i}{r}}|u_1\rangle$$

- We can generalize this a bit more by multiplying by s to give:

$$|u_s\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{\frac{2\pi isk}{r}}|a^k mod\ N\rangle$$

$$U|u_s\rangle = e^{\frac{2\pi is}{r}}|u_1\rangle$$

- This gives us a unique eigenvector for each value of s, 0<s<r-1

Shor's Algorithm

- If we sum all of these eigenvectors the phases for all the computational basis cancel out, except for $|1\rangle$
- That is we have:

$$\frac{1}{\sqrt{r}}\sum_{s=1}^{r-1}|u_s\rangle = |1\rangle$$

- If we do a quantum phase estimation with the eigenvector $|1\rangle$ we will measure the phase:

$$\phi = \frac{s}{r}$$

Shor’s Algorithm

- We select a random s, run the quantum phase estimation algorithm and then use the continued fractions algorithm on the result to retrieve r
- This value of r gets plugged into step 5 of Shor’s algorithm and we hope to retrieve a factor of our number
- This won’t always be the case, if it fails we need to choose a different random number s and repeat the process

Shor's Algorithm

- There is still one detail, we have that:

$$U|y\rangle = |ay \; mod \; N\rangle$$

- And we need to calculate a series of U^{2^j}
- These are the gates in our quantum phase estimation algorithm
- This is going to be a complicated circuit if we aren't careful
- It turns out that there are efficient ways of doing this, but we won't look at them now, maybe later

Grover's Algorithm

- Grover's algorithm is used to search unstructured data, that is data that isn't sorted
- In the case of an unsorted list of N items, and an item that we are searching for, w, the best we can do on a classical computer is to go through the list, item by item, looking for w
- In the worst case we will need to examine N items, and on average we will need to examine N/2 items
- Grover's algorithm reduces this to $\sqrt{N}$

Grover’s Algorithm

- At first this doesn’t look like much of an improvement but for large lists its sizeable
- Consider a list of 10,000 items, a classical search will need 5,000 steps while Grover’s algorithm only needs 100
- This is one of the reasons why Google is interested quantum computers
- Grover’s algorithm uses a trick called amplitude amplification, which is a generally useful technique

Grover's Algorithm

- Grover's algorithm is based on the idea of an oracle that tells use whether we have found the item
- Again assume w is the item we are searching for, the oracle Grover uses is:

$$U_w|x\rangle = \begin{cases} |x\rangle \; if \; x \neq w \\ -|x\rangle \; if \; x = w \end{cases}$$

- In the case of N=8 this produces the matrix on the next slide when w = 5

Grover's Algorithm

$$U_\omega = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \leftarrow \omega = 101$$

Grover's Algorithm

- Another possible oracle is:

$$U_w|x\rangle = (-1)^{f(x)}|x\rangle$$

- Where

$$f(x) = \begin{cases} 0 \; if \; x \neq w \\ 1 \; if \; x = w \end{cases}$$

$$U_\omega = \begin{bmatrix} (-1)^{f(0)} & 0 & \cdots & 0 \\ 0 & (-1)^{f(1)} & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & (-1)^{f(2^n)} \end{bmatrix}$$

Grover's Algorithm

- Start with an unsorted list of $N=2^n$ items and we are looking for the item w
- We don't know where w is, so all locations are equally likely to contain it
- We can express this as the following superimposed state including all the locations in equal amount

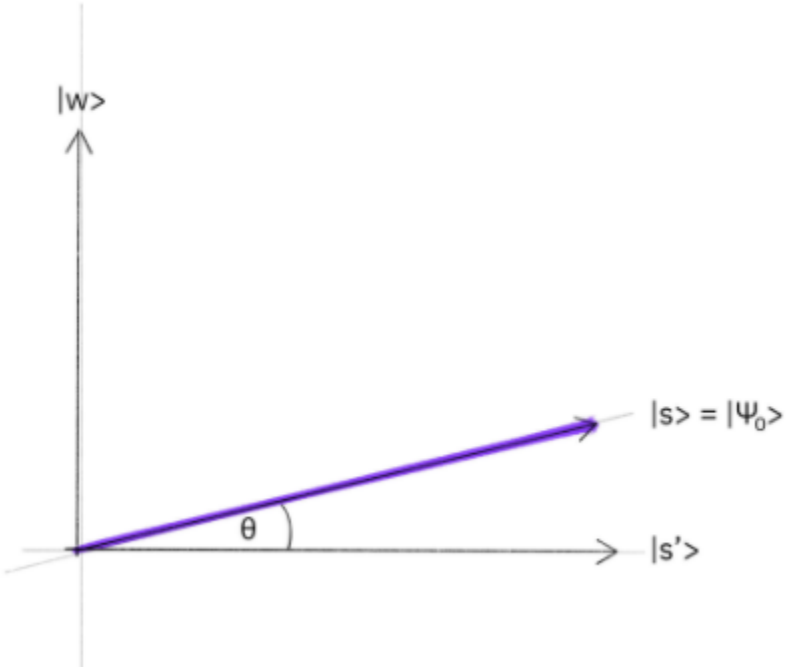
$$s = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

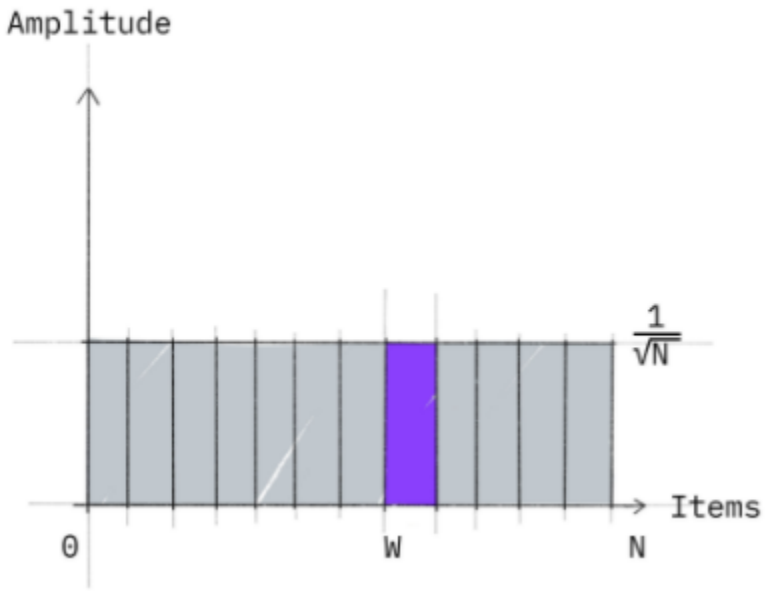
- Where each item has the probability 1/N and amplitude $1/\sqrt{N}$

Grover's Algorithm

- Grover's algorithm increases the probability of the location w, and decreases the probabilities of all the other locations
- It repeats this process until we are certain of the location of w
- We start with two vectors, |s> and |w> that span $\mathbf{C}^n$, but they aren't orthogonal
- We create a new vector |s'> by subtracting |w> from |s> and rescaling
- We have that |s'> and |w> are orthogonal

Grover’s Algorithm





Grover's Algorithm

- We construct |s> in the usual way by applying Hadamard gates:

$$|s\rangle = H^{\otimes n}|0\rangle^{\otimes n}$$

- From the previous slide we also have

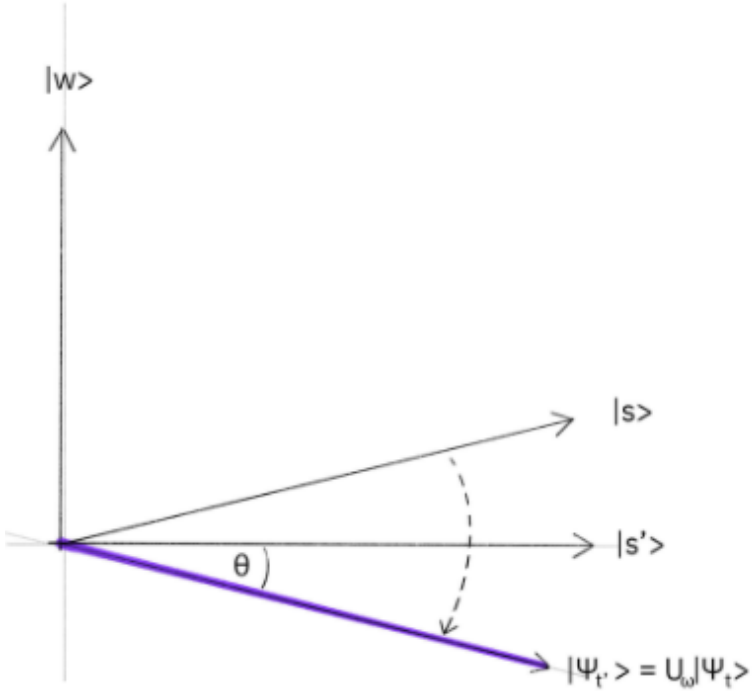
$$|s\rangle = sin\theta|w\rangle + cos\theta|s'\rangle$$

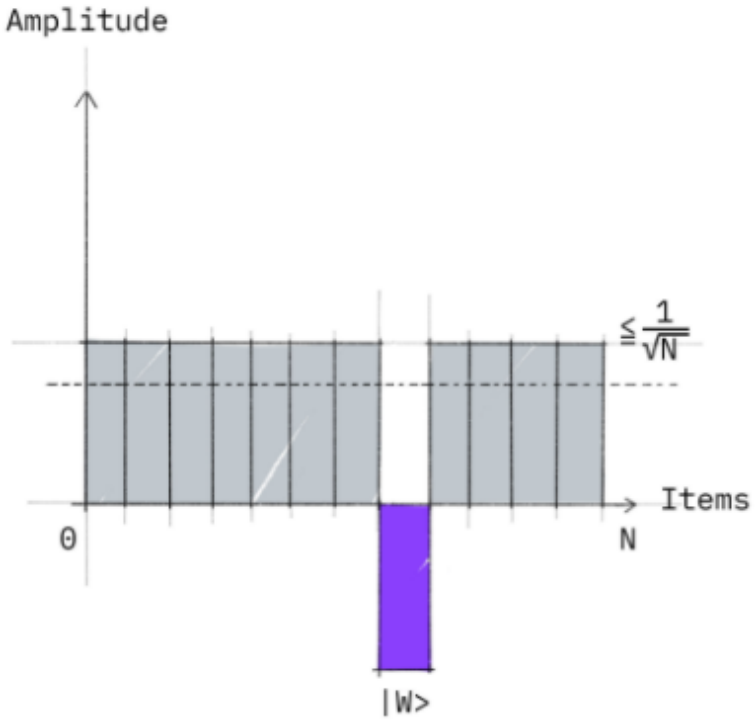
- Where

$$\theta = \arcsin(s|w) = \arcsin(\frac{1}{\sqrt{N}})$$

- Now we apply our oracle U_w to |s> as shown in the next slide

Grover's Algorithm





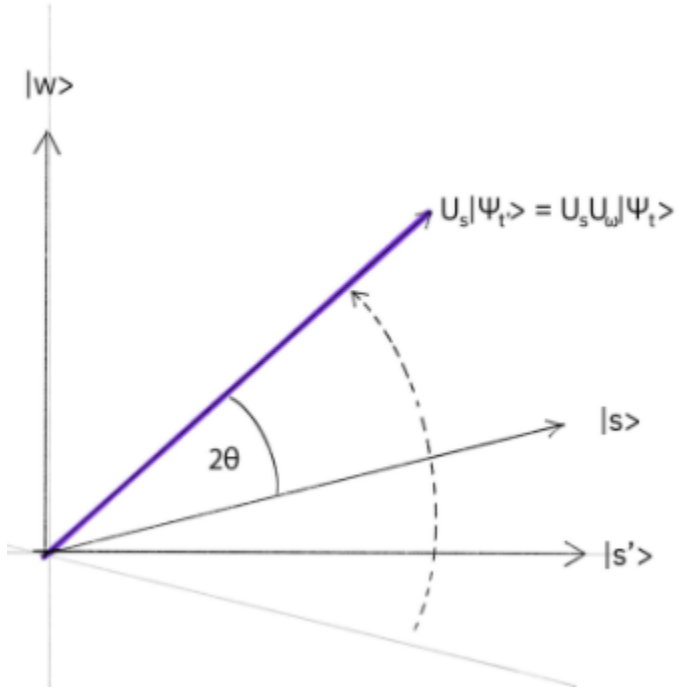
Grover's Algorithm

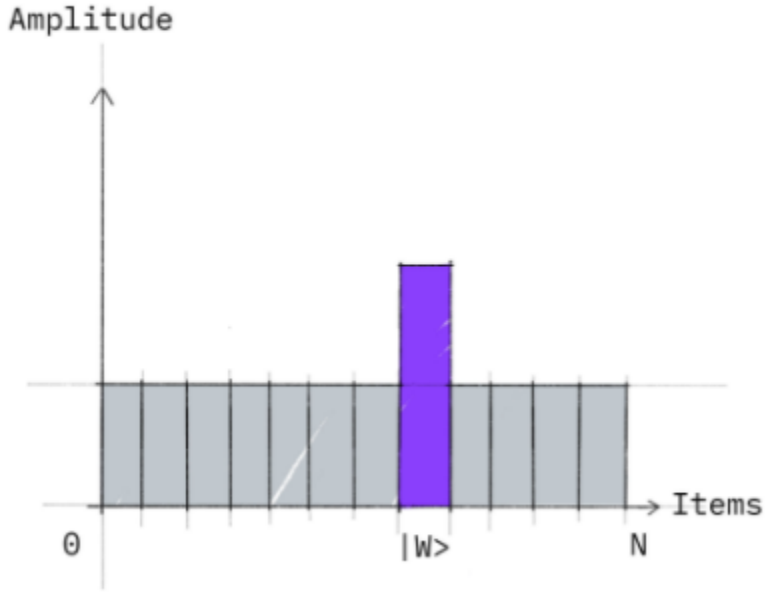
- This is a reflection about the s' axis
- Note the item with w now has a negative amplitude, and the average amplitude has been reduced below $1/\sqrt{N}$
- That is, we have reduced the probability of the locations without w
- Now we apply another reflection U_s given by:

$$U_s = |s\rangle\langle s| - 1$$

- This gives the result on the next slide

Grover's Algorithm





Grover's Algorithm

- Note that the location with w now has a higher probability
- After we have repeated the process t times we will be in the following state:
- $|\psi_i\rangle = (U_s U_w)^t |\psi\rangle$
- How many times do we need to repeat these steps, in turns out that $\sqrt{N}$ is good enough
- We may converge before then, but by that point we will have our answer

Summary

- We've examined two major techniques for constructing quantum algorithms:
  - Quantum phase estimation
  - Amplitude amplification
- The first technique was a major part of Shor's algorithm, which can crack cryptographic codes
- The second techniques is the heart of Grover's algorithm
- We've now seen the major algorithms, it's time to move on to hardware