Grover’s Algorithm

Searching for quantum advantage with practical applications

Outline

History and motivation of the algorithm

- Original formulation – “database” search
- Practical applications
  - Factoring
  - Clustering
- Speedups compare to classical

Algo details

- Formalism (unstructured search, oracle)
- Oracle = Reflection
- Reflection = Amplification

Final remarks

Whose exam is this??

An exam was submitted with only the ID number (but no name!)

To find out who the exam belong to:

- Student directory – input student's name returns their ID number
- Go through all the students until finding the ID number matching the one on exam?

… 😵‍💫 very slow!

Photo credit: https://phdcomics.com/

The Database Search Problem

This is the original “Phone-directory” problem considered by Lov Grover in 1996

“A fast quantum mechanical algorithm for *database* search”

- Search space
  - All students taking the class
  - Could also be abstract in general: all possible ways to partition a set into two groups.
- A way to quickly check if the input is desired solution (the “oracle”)
  - Clicking into their page and checking if their ID matches the unnamed exam
- How to find who the exam belongs to with the least number of clicks?

Quantum Search: Grover's Algorithm

A general-purpose search algorithm applicable to problems that can be solved by a classical brute-force search

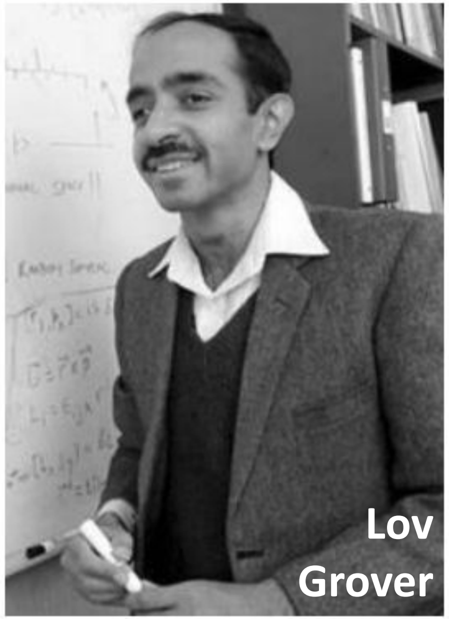
One of the most famous quantum algorithms (after Shor's Algorithm)

In reality, "Database" is a misnomer–
There is often no data!!

- Quantum systems are bad for persistent storage
- Costs of entering data outweigh benefits of QAlgs

Usually used to search *all possible solutions*

Photo credit: https://medium.com/@sakibkhan/grovers-algorithm-for-searching-a-quantum-approach-1488b325151f



Examples: Searching for a good solution

In reality, suitable for abstract problems that does not require data storage

- ***Identifying Good Models:*** Finding a good classification and clustering models for complicated data sets
  - Inputs/Search Space: model parameters
  - Good solutions/Oracle: models that have a good prediction rate
- ***One-Way Functions:*** A function with good solutions that are computationally difficult to generate but easy to verify; e.g., solutions to a sudoku puzzle
  - Inputs: Arbitrary numbers filling the sudoku puzzle
  - Good solutions/Oracle: solutions that obey the rules
- ***Etc…***

Grover’s Algorithm Target Application Class

- *Large search space* - Lots of potential solutions, few *good* solutions
- *Quick oracle* - Can quickly check whether a solution is good
- *Unstructured search* - Checking whether one solution is good does not give you information on potential solution
  - No good heuristics of narrowing down the search space

→ **Best Quantum Advantage when the best classical approach is brute-force**

Factoring a large number

Very computationally expensive to calculate the solution

Lots of possible solutions

Grover's approach:

- Given a cryptographic key, try all pairs of numbers
- Check to see if they satisfy constraints
    - Is their product the known key?
    - Are they both prime?

Compared to classical approach:

There are heuristics to narrow down numbers, so classical is not brute force

Grover's can also take advantage of quantum algorithms

Clustering and classification

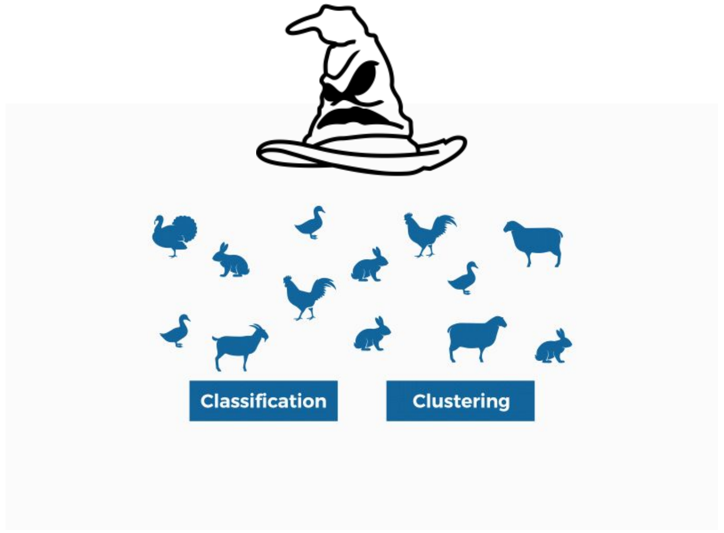
Given a set of items, how do we separate them into groups of similar items, and predict where a new item belongs?

Real world applications:

- Image recognition – cat / dog pictures
- Identifying defective products from good ones
- Automatic sorting of recycled items
- Medical diagnostics

  …

- Sorting Hat from Harry Potter



Quantum solution to clustering on difficult data sets

Given n points, there are $N = 2^n$ possible ways to assign them to two clusters

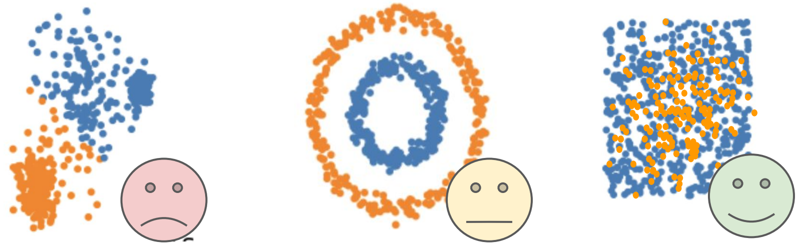
Grover's approach:

- A clustering solution can be mapped to an n-bit number, where the ith digit tells us which cluster point i is assigned to
- Check to see if the solution is good
    - Oracle counts the number of correctly labeled points
        - Good solution if this number is above a certain threshold
    - Takes n = O(log(N)) time to implement!

Compared to classical approach:

ML algorithms exist for some data - quantum only useful

on data sets that are hard for existing classical algorithms



A Quantum Searching Solution: Grover’s Algorithm

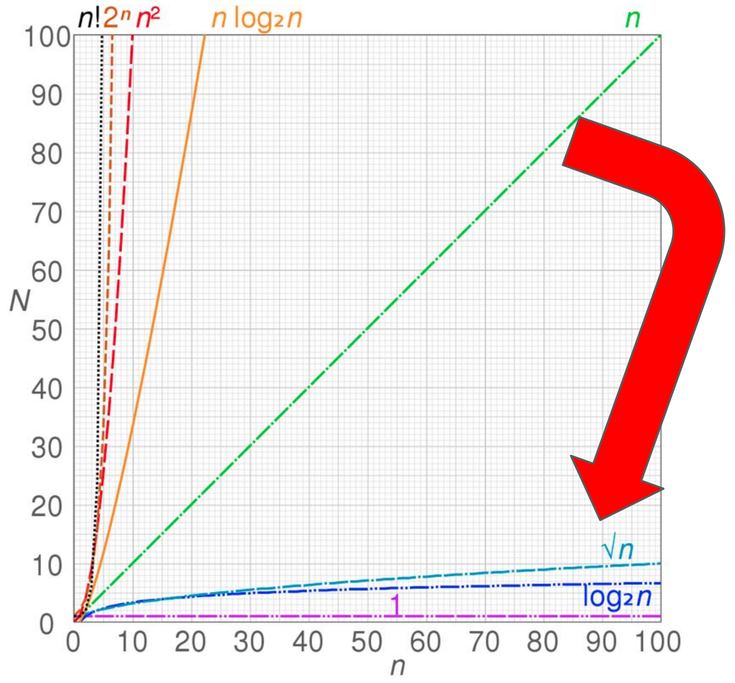
Takes advantage of qubit superposition and phase interference to improve unstructured database search from O(N) to O(√N)

- Note: ‘Big O’ notation provides insight on the upper bound of the resources an algorithm needs

In the clustering example–suppose we have 10 items to classify, to find the best solution would require us to search over 2^10 = 1024 different clusterings!

Meanwhile, the same problem can be solved with Grover’s algorithm on a 10-qubit quantum computer using Hadamard gates, and ***1023*** guesses (worst case classically) becomes √***1024 = 32*** circuit layers!

This is a ***quadratic speedup!***



Outline

History and motivation of the algorithm

- Original formulation – “database” search
- Practical applications
  - Factoring
  - Clustering
- Speedups compare to classical

Algo details (Do the following slides on board)

- Formalism
- Understanding oracles as reflections rotations
- Using rotations to move closer to the answer

Final remarks

The Unstructured Search

Let’s now formally define the problem…

- Suppose we are given a function f on inputs $x \in \{1,...,N\}$ and outputs either 0, indicating a bad solution, or 1, indicating a good solution
- We’ll make two assumptions:
  - Let’s assume there’s only one good solution. That is, there exists a unique $w \in \{1,...,N\}$ such that $f(w) = 1$, and $f(x) = 0$ for all $x \neq w$. This is the hardest case to solve!
  - Let’s also say, for convenience, that $N = 2^n$ for some integer n. This ensures we can load the problem into the quantum computer properly
  - E.g.: f can be the loss function of a clustering model

The goal is to find w with the least number of evaluations of f(x)

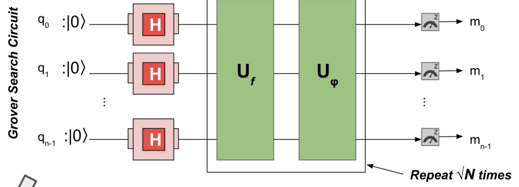
Amplitude Amplification

Grover's algorithm implements an ***amplitude amplification*** to increase the probability of observing the correct answer (the "good solution")

- **INCREASES** probability amplitude associated with answer |w⟩
- **DECREASES** all other probability amplitudes

BALANCED SUPERPOSITION

$$|\psi\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$



E.g.,|w⟩= |11⟩

NEW STATE WITH |11⟩AMPLIFIED

$$|\psi'\rangle = \frac{1}{\sqrt{12}}|00\rangle + \frac{1}{\sqrt{12}}|01\rangle + \frac{1}{\sqrt{12}}|10\rangle + \frac{\sqrt{3}}{2}|11\rangle$$

Note:

$$1 = 3 * \left(\frac{1}{\sqrt{12}}\right)^2 + \left(\frac{\sqrt{3}}{2}\right)^2$$

This is done through a sequence of reflections around the good solution

The Unstructured Search - Quantum Oracle

The quantum oracle U_f is given as a phase oracle: $U_f\,|x\rangle = (-1)^{f(x)}\,|x\rangle$; that is:

$$U_f\,|x\rangle = -\,|x\rangle \quad \text{if } x = w \text{ is a good solution,}$$

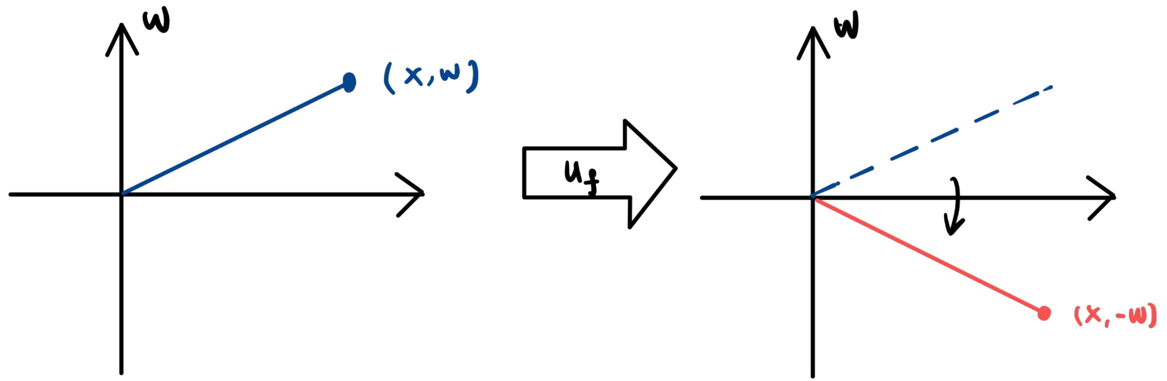
$$U_f\,|x\rangle = |x\rangle \quad \text{if } x \neq w$$

- The conventional oracle is given as $U_f^*|x\rangle \otimes |y\rangle = |x\rangle \otimes |y \oplus f(x)\rangle$; we can turn it into a phase oracle sing a $|-\rangle$ ancilla and phase kickback, similar to what is done in the Deutsch-Jozsa Algorithm
- We always assume the underlying function, f, is easy to compute. U_f^* computes f and can write the result to a $|-\rangle$ ancilla using CNOT gates
  - In the clustering example, the oracle calculates the success rate of a clustering and compares it to a set threshold to determine whether a solution is good (can apply to a superposition!)
  - Note it is OKAY to take O(poly(n)) time to implement U_f, since n = O(log N)

The Unstructured Search - Quantum Oracle

The quantum oracle U_f given as a phase oracle: $U_f\,|x\rangle = (-1)^{f(x)}\,|x\rangle$

- The action of U_f on a state is a ***reflection*** about the superposition of bad solutions

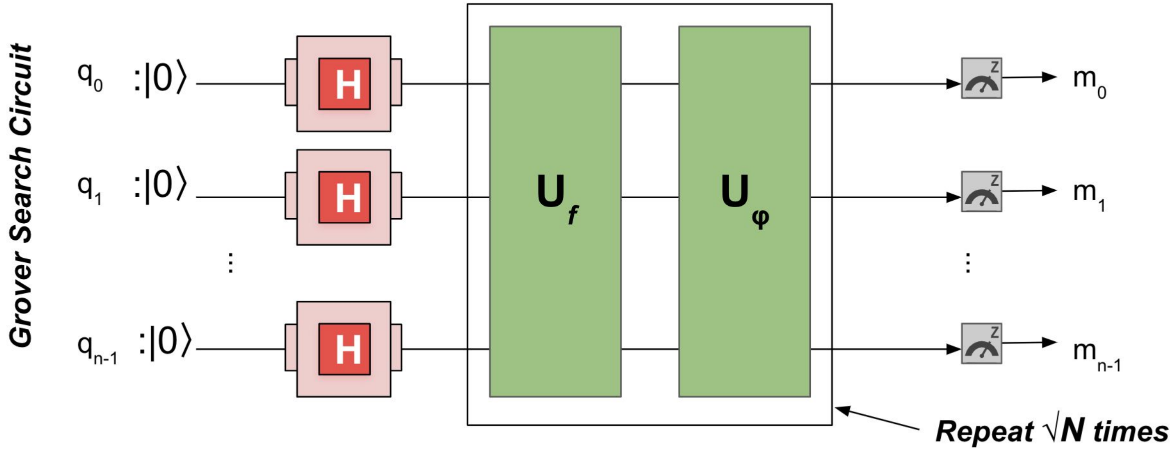


- Amplitude amplification is just a sequence of clever reflections that brings an initial guess closer to the good solution!

Amplitude Amplification as Reflections

Starting from a uniform superposition, the Grover's algorithm works by repeatedly reflecting our guess around the bad solutions $|w^{\perp}\rangle$ and our first guess $|s_o\rangle$.

Each iteration increases the amplitude of $|w\rangle$ in our guess $|s\rangle$ by O(1/√N); our guess $|s\rangle$ turns into $|w\rangle$ after √N iterations!

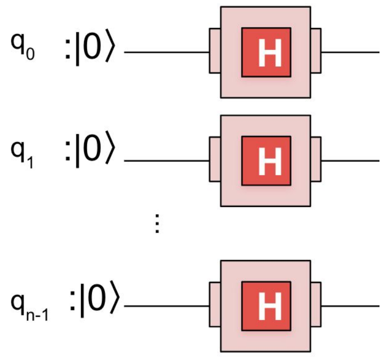


We'll see how these reflections brings our guess close to a good solution…

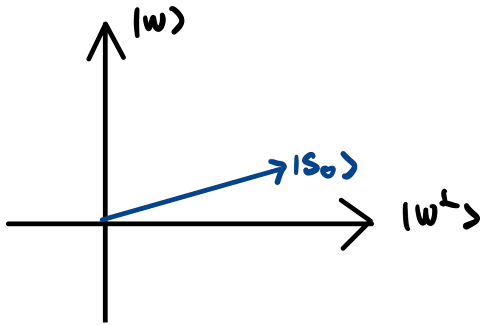
Initial Guess

We begin with an initial guess, the equal superposition over all possible outcomes

- Let's call the initial guess $|s_0\rangle = 2^{n/2}(|00..0\rangle + |00..1\rangle + \ldots + |11..1\rangle)$
- Since only one possible outcome (out of 2^n many!) corresponds to a good solution, the initial guess is very far away from $|w\rangle$ and close to $|w^\perp\rangle$, the superposition of all bad solutions

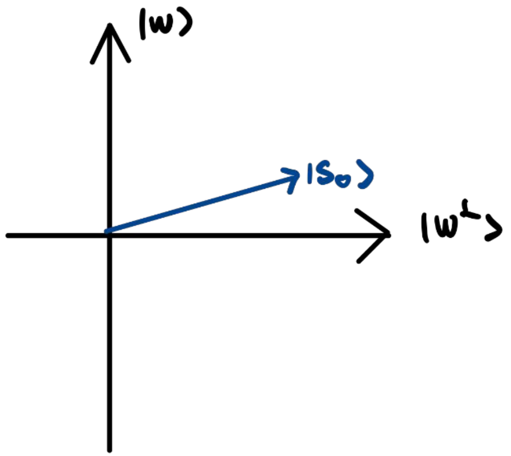


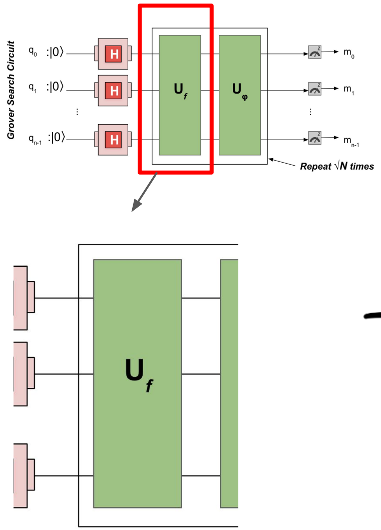
We can visualize $|s_0\rangle$ and $|w\rangle$ as:

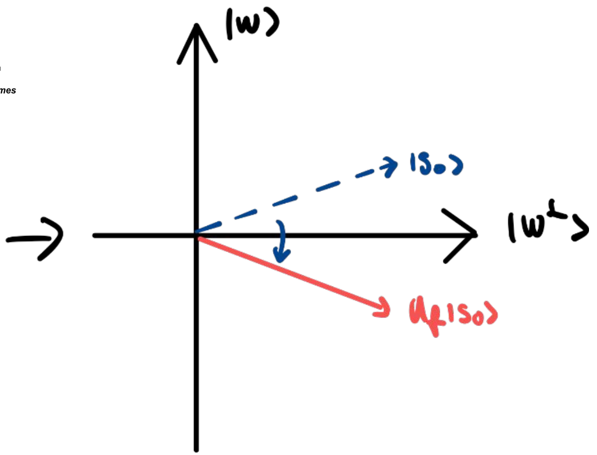


Step 1: Reflection about $|w^{\perp}\rangle$

We have access to the Grover oracle U_f that flips the phase for only $|w\rangle$. As we've seen before, this corresponds to a reflection around $|w^{\perp}\rangle$



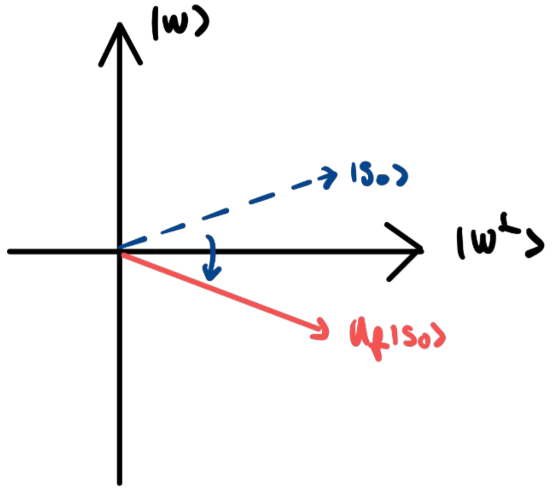


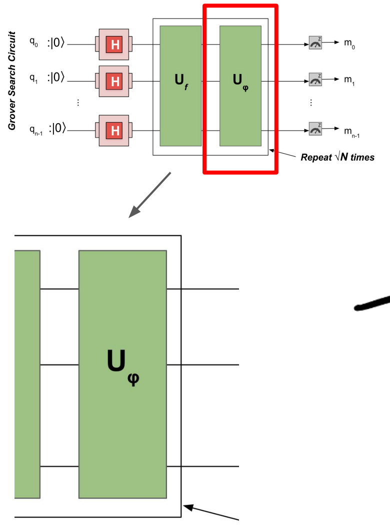


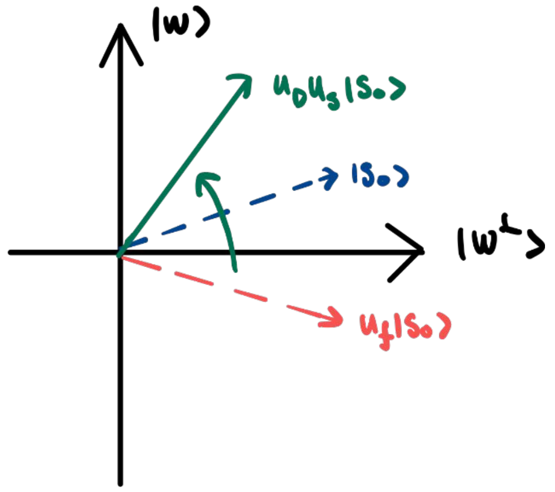
Step 2: Reflection about $|s_o\rangle$

Next, we can apply the Diffusion Operator U_D that reflects our guess about the uniform superposition $|s_o\rangle$

- This oracle is easy to implement since $|s_o\rangle$ is easy to prepare



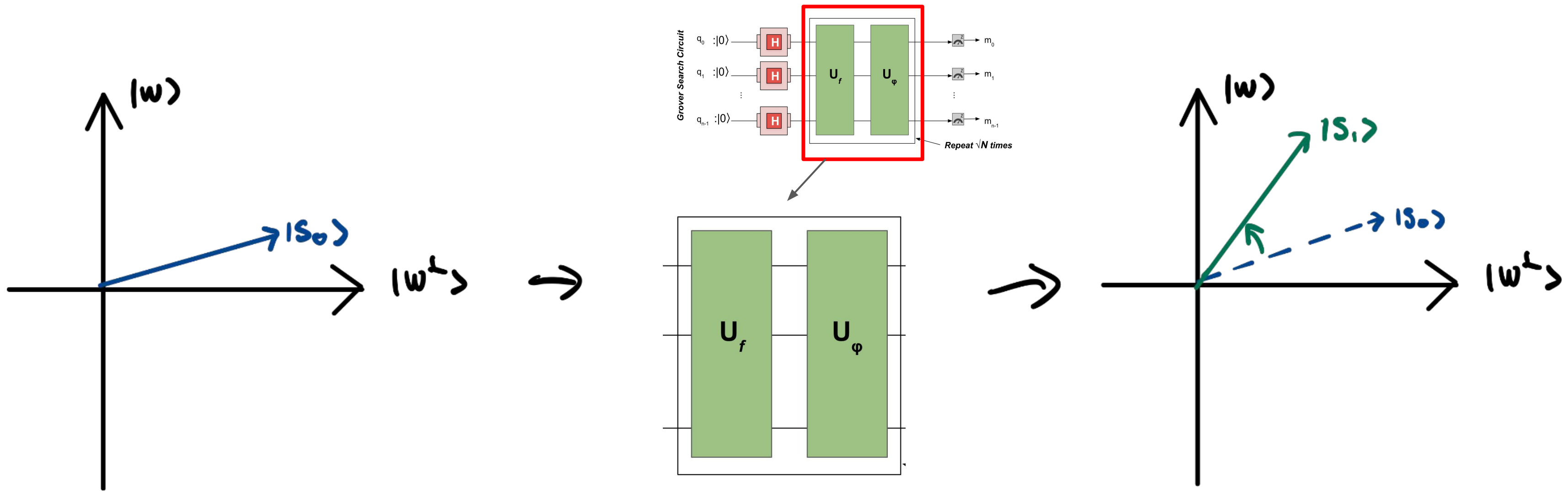




Step 1+2: Overall effect

Note the overall effect the two steps brings our initial guess $|s_o\rangle$ closer to $|w\rangle$

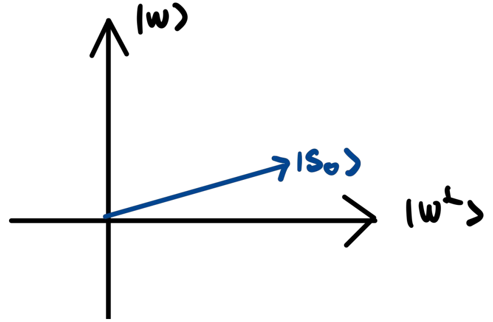
- In other words, the amplitude of $|w\rangle$ in $|s_o\rangle$ is "amplified" by around $1/\sqrt{N}$

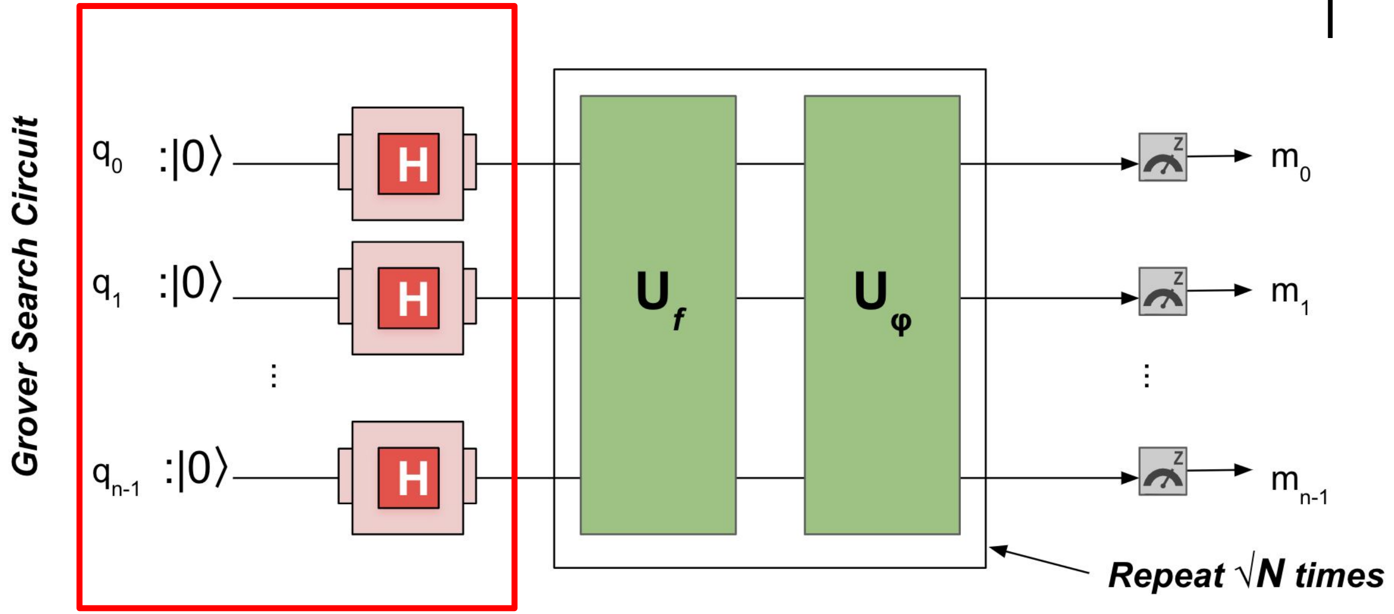


The Grover Algorithm

Putting everything together:

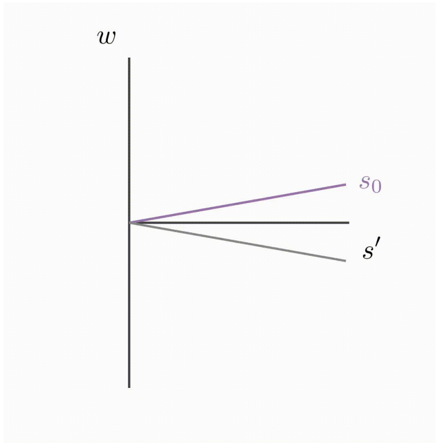
- Initialization : Prepare uniform superposition $|s_o\rangle$

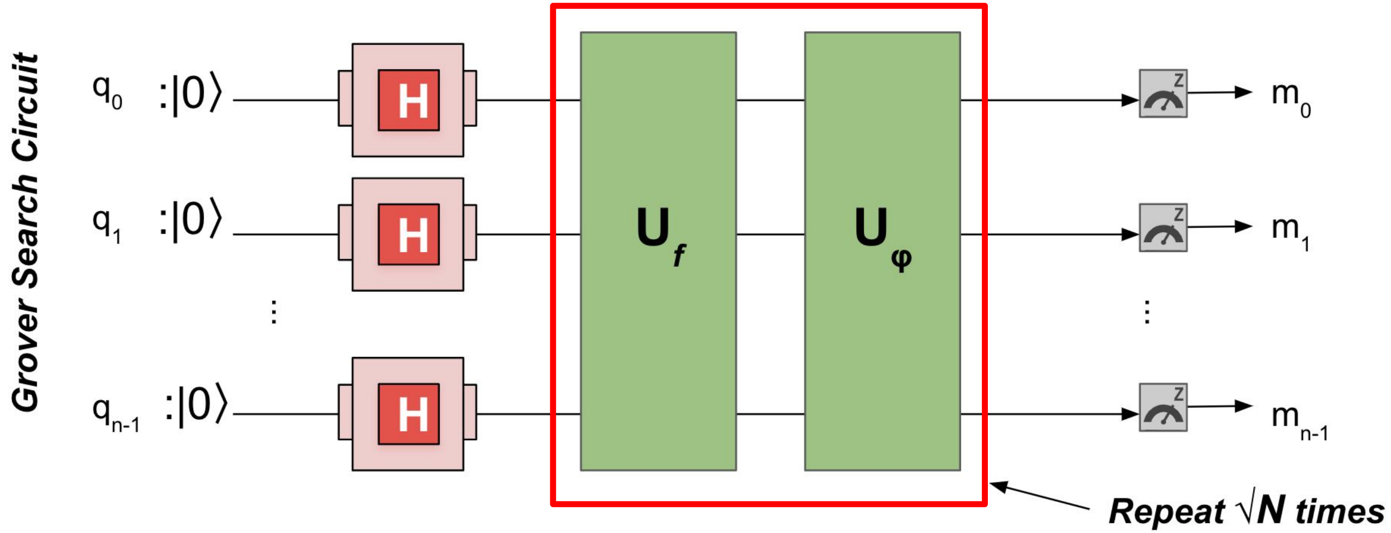




The Grover Algorithm

- Grover Rotation (Repeat √N times):
  - Reflect $|s_i\rangle$ around $|w^{\perp}\rangle$, obtain $|s'_i\rangle = U_f|s_i\rangle$
  - Reflect $|s'_i\rangle$ around $|s_o\rangle$, obtain $|s_{i+1}\rangle = U_D|s'_i\rangle$
  - Magnitude of $|w\rangle$ is magnified by around 1/√N

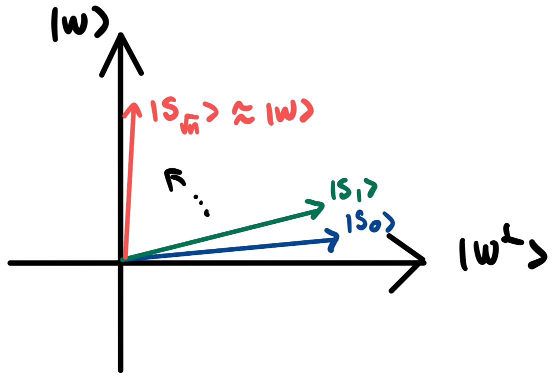


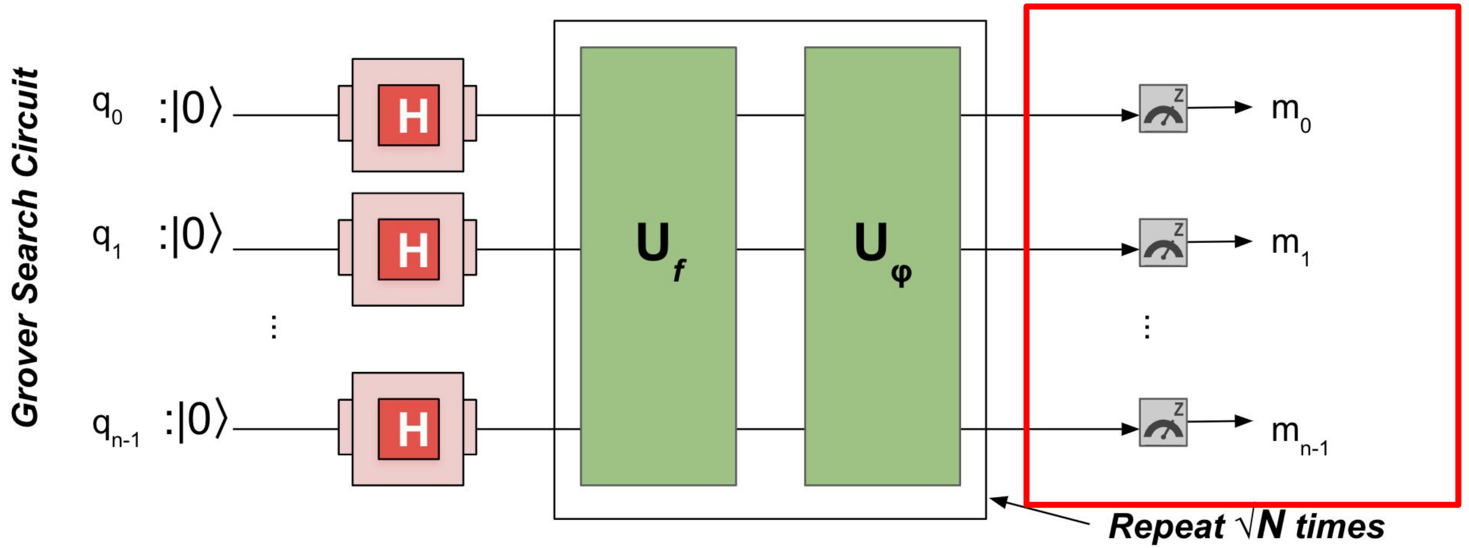


The Grover Algorithm

- Measurement

  With high probability, the outcome reveals $|w\rangle$





Additional Notes for Grover’s Algorithm

- There are a few ways to implement for the Grover oracle, U_f, but the mechanics for the search process stay the same
- We searched for one item, but variations of Grover’s Algorithm can search for multiple database targets
  - For t possible matches, the oracle is repeated $\sqrt{(N/t)}$ times
- Partial quantum search is also possible!
  - Example: Determine if a student’s final grade falls in the lower 25%, 25–50%, 50–75% or 75–100% percentile without caring about exact ranking
- Can solve optimization problems using Grover as a subroutine
  - We can guess the optimal value using binary search, and apply Grover’s algorithm to iteratively find good solutions that achieves the optimal values

A Classical Attempt

Finding a good solution can be challenging for classical computers on **problems with limited structure in potential solution space** (calculating one solution does not provide information relevant to other potential solutions)…

- Best classical approach would be a brute-force search!
- When there's only a constant number of good solutions, the best we can hope for is O(n) guesses on average, or around n guesses in the worst case

Generic Example: The Guessing Game

Let’s use a convenience example to compare the runtime guarantees. We’ll consider a guessing game:

Pick a random number between 1 & 1024 to be our good solution

$$\underbrace{\mathbf{1, 2, \dots, 1023, 1024}}_{\textit{1024 total options}}$$

Worst case: requires 1023 calls to the classical oracle

- Only one correct answer
- Incorrect guesses provide no information

This is the hardest example of an ***unstructured search***

Simple Example: The Search Problem

In the worst case, it would take O(N) times to locate an item in a database of size N

If random access and pre-sort are added to implement binary search on ordered data, search time reduces to O(log(N))

$$\underbrace{1, 2, \dots, 1023, 1024}_{\textit{1024 total options}}$$

Worst case: requires 1023 calls to the classical oracle

Remember, sorting procedures still require time!

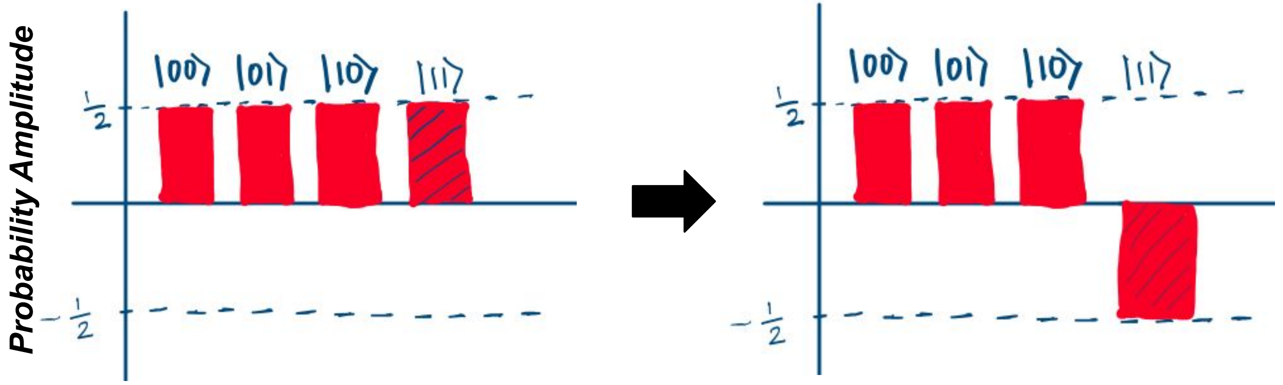
Search time reduction isn't free!

Amplitude Amplification Procedure

Step 1: Start with a balanced superposition, and assign a phase of -1 to the chosen ket, $|11\rangle$

BALANCED SUPERPOSITION

$$|\psi\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

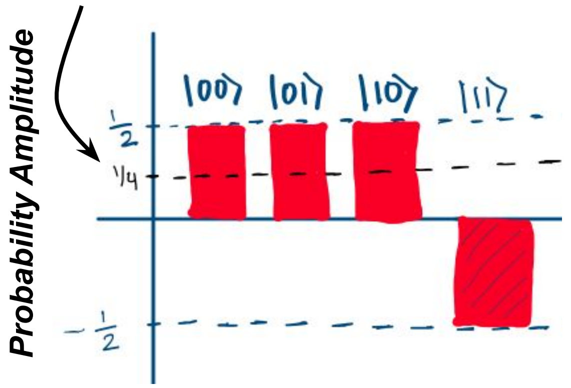


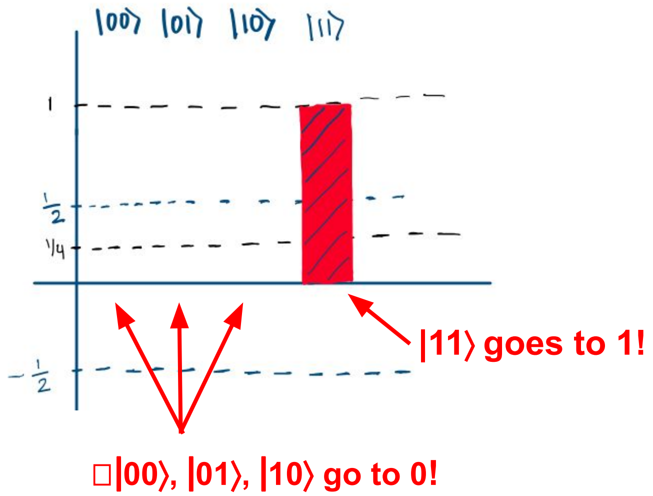
We will call the matrix that implements the phase-flip U_f

Amplitude Amplification Procedure

Step 2: Invert all probability amplitudes about the mean (average)

Average(0.5, 0.5, 0.5, -0.5) = 0.25





We will call the matrix that inverts around the mean U_φ

PRACTICE:

In the previous example, what matrix could be used to assign a phase of -1 to the chosen ket, $|11\rangle$?

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

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

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

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

PRACTICE:

In the previous example, what matrix could be used to assign a phase of -1 to the chosen ket, $|11\rangle$?

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

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

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

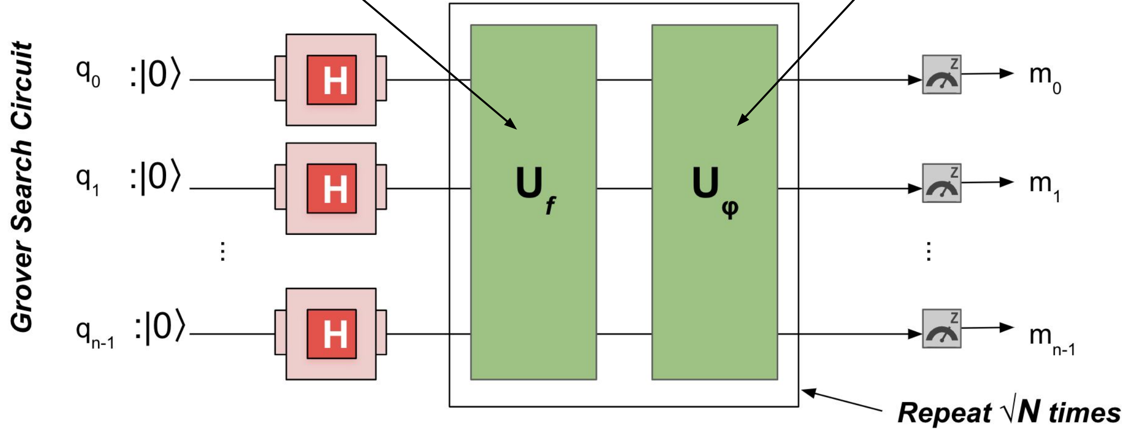
$$\boxed{\text{d.}\quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}}$$

Amplitude Amplification: Two Parts

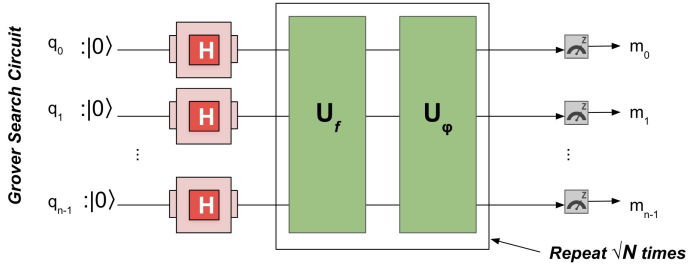
*The **Oracle** flips the sign of the desired search element*

*The **Diffusion Operator** inverts amplitudes about the mean*

How does Oracle 'know' what to look for?



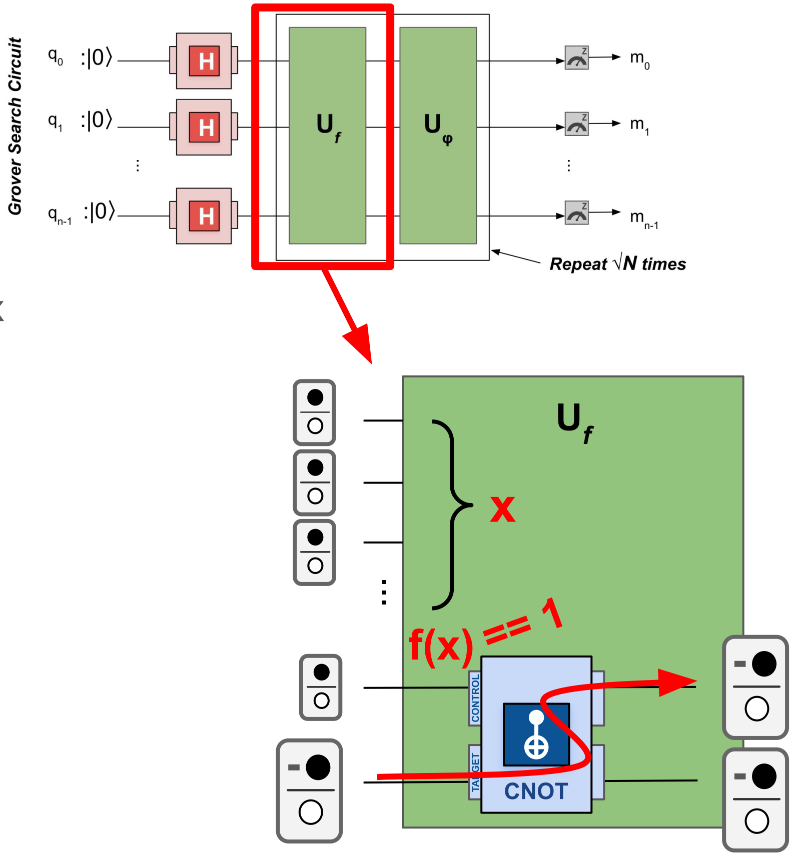
Building the Grover Oracle: Bird’s Eye View



1. Develop classical function, f(x), that outlines search criteria...f(x) == 1 when x is the solution
   - ‘Find an input x that’s a factor of 25’
   - ‘Starting with these numbers, find the arrangement in a 9x9 grid where every value appears once in a region, a row, and a column’
   - ‘Find the optimum route where I visit 300 stops only once’

f(x) = 0, if input x doesn’t satisfy criteria ❌
1, if x input satisfies criteria ✅

Building the Grover Oracle: Bird's Eye View



2. Create a reversible circuit that calculates f(x) with input x

- Reminder: we can create a REVERSIBLE circuit from an IRREVERSIBLE circuit with ancilla!
- Reversible circuit generators help create quantum circuits from classical computation

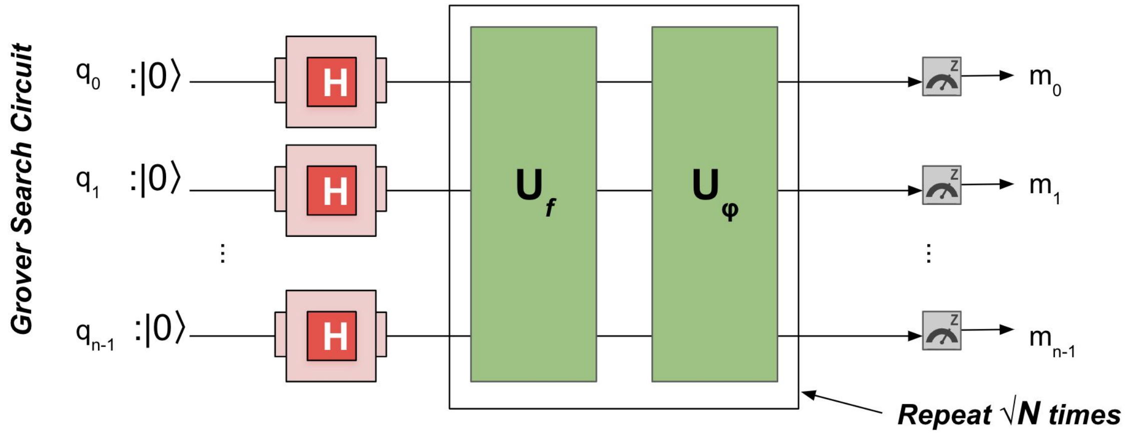
3. Add an additional ancilla with negative phase that introduces PHASE KICKBACK when f(x) = 1

- Object of search gets a phase of -1!

Grover's Search Mechanics

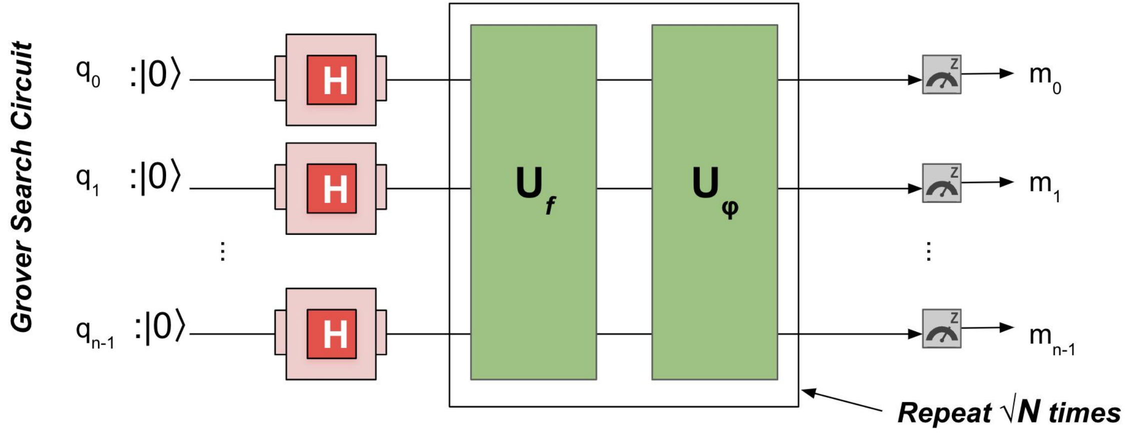
1. Identify your search problem and the data (N items) you wish to search
2. Find the smallest number n that satisfies $N \leq 2^n$

   Input x represented with n qubits in the search algorithm … superposition via H allows us to 'look' at all values at once



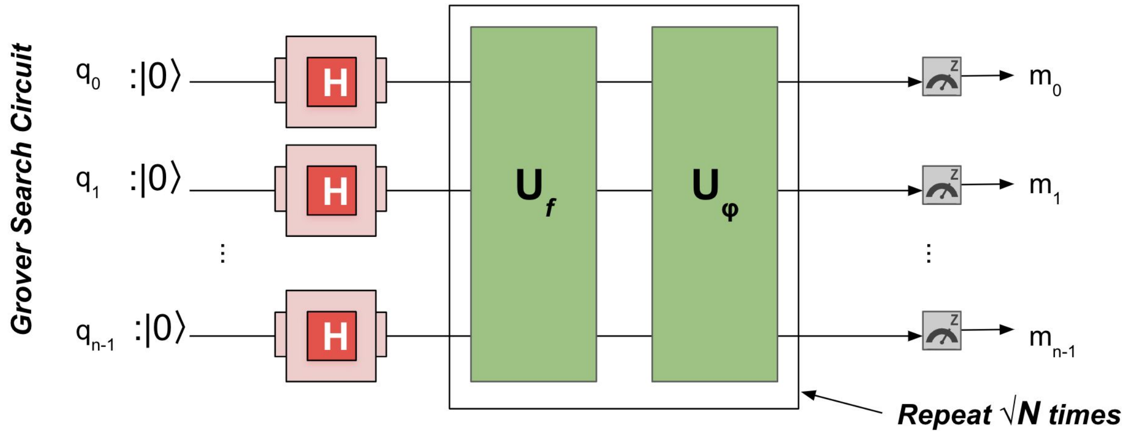
Grover's Search Mechanics

3. Construct an **Oracle, f** and a gate/matrix $\mathbf{U}_f$ that flips the sign of the object of search
4. Run $\mathbf{U}_f$ that flips the sign of the object of search followed by $\mathbf{U}_\varphi$ the circuit that inverts around the mean (we call this the **Grover Diffusion Operator**). Repeat this step **√N times**



Grover's Search Mechanics

5. Measure and read off the state with an encoding that corresponds to the desired item in the database. Map the resulting state back to the data



This material is based upon work supported by the National Science Foundation under Grants No. 1730088 and No. 1730449. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Understanding the Oracle with IBM QE

We will build the circuit for a two-qubit search for 11

More examples in the qiskit textbook:
https://qiskit.org/textbook/ch-algorithms/grover.html#2qubits

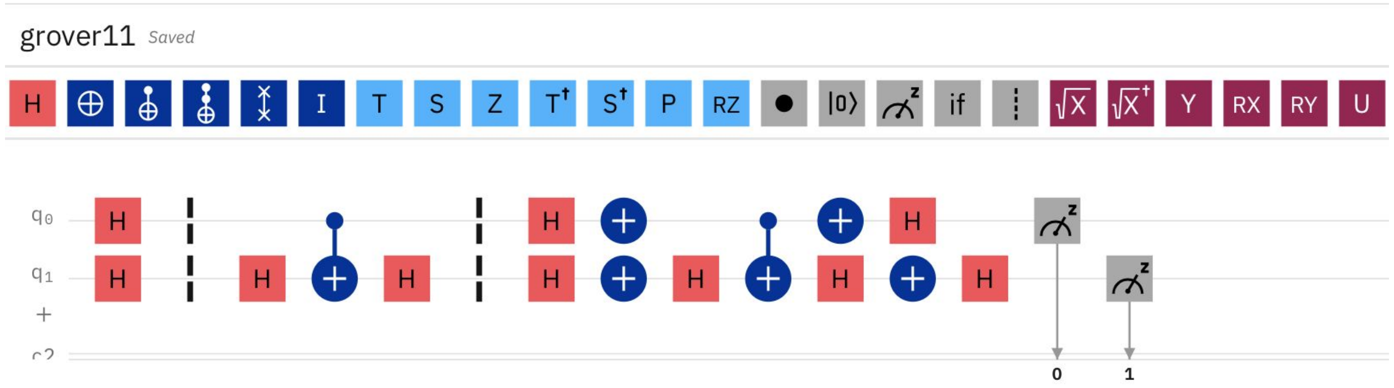
Grover’s QASM

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];

h q[0];
h q[1];
barrier q[1],q[0];
h q[1];
cx q[0],q[1];
h q[1];
barrier q[1],q[0];
h q[0];
h q[1];
x q[0];
x q[1];
h q[1];
cx q[0],q[1];
x q[0];
h q[1];
h q[0];
x q[1];
h q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

Grover's circuit

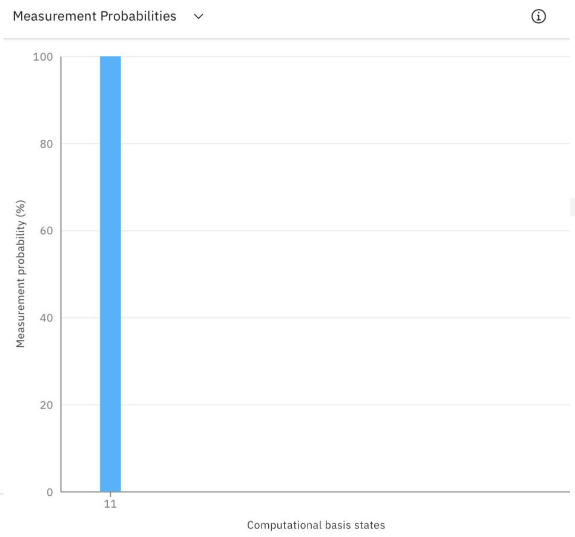


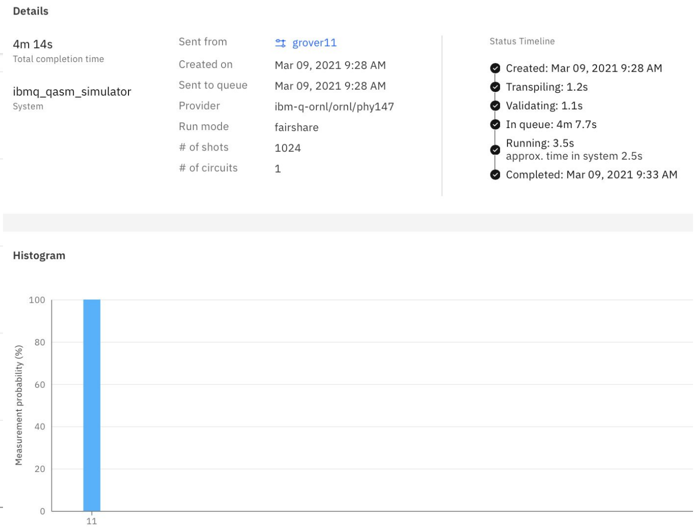
First Stage: Create Superposition

Second Stage: Implement Oracle that flips sign

Third Stage: Implement Grover's Diffusion Operation

Grover's circuit




Details

4m 14s
Total completion time

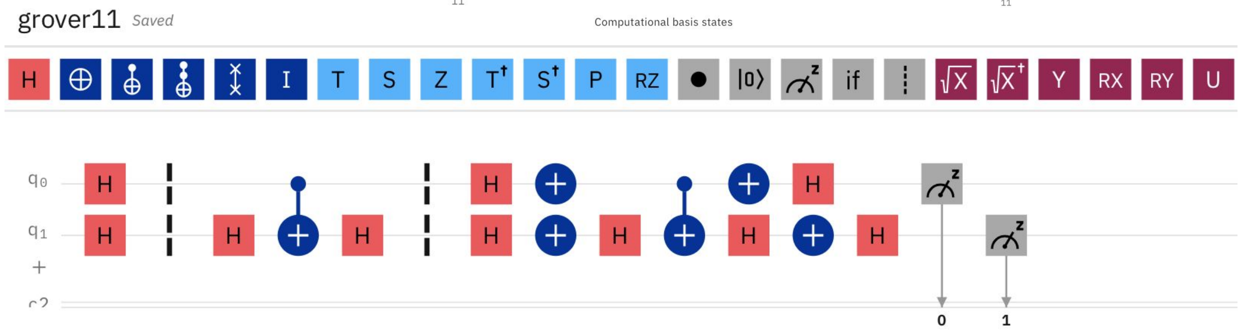
ibmq_qasm_simulator
System

Sent from	grover11
Created on	Mar 09, 2021 9:28 AM
Sent to queue	Mar 09, 2021 9:28 AM
Provider	ibm-q-ornl/ornl/phy147
Run mode	fairshare
# of shots	1024
# of circuits	1

Status Timeline

- Created: Mar 09, 2021 9:28 AM
- Transpiling: 1.2s
- Validating: 1.1s
- In queue: 4m 7.7s
- Running: 3.5s
  approx. time in system 2.5s
- Completed: Mar 09, 2021 9:33 AM

Histogram


grover11 *Saved*

Notes - Diana, Pranav, Kate discussion!

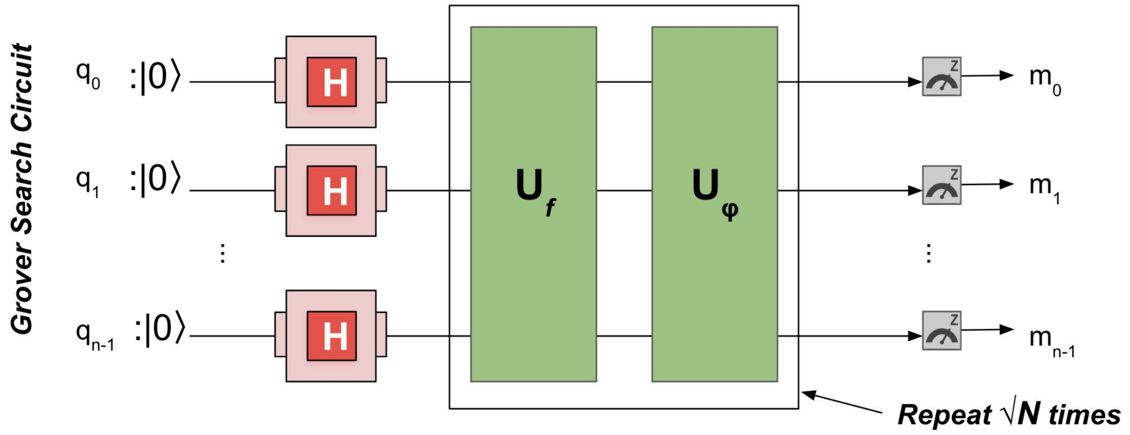
-How does a phase oracle exist in reality?
    Mike and ike (pg 253) has a pic for grovers search
-searching for 11011...what is this?
-if you have your answer, why search for it
-when building the oracle, it is related to the problem, but it does not know the answer….solution CHECKER
    -its a classical checker of 0 if no/1 if yes (irreversible), use reversible logic generator to produce oracle
    - oracle introduces phase kickback (-1)^f(x) to put phase on true statement!
Qiskit textbook added some good detail: https://qiskit.org/textbook/ch-algorithms/grover.html

OLD

Grover’s Search Mechanics

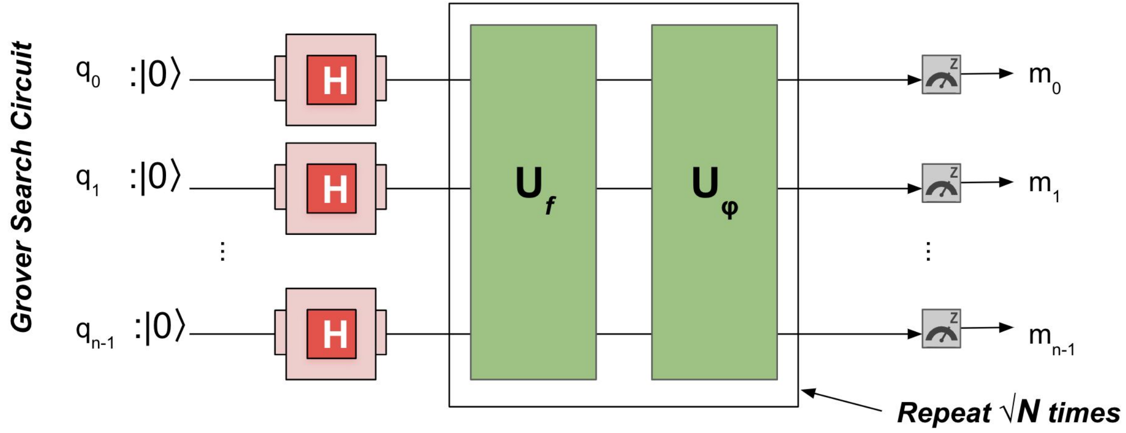
1. Identify the data (N items) you wish to search
2. Find the smallest number n that satisfies $N \leq 2^n$

   A total of n qubits will be used in the search algorithm (each item has a bit string ID/already represented within states of qubits), and they will be put into superposition with H before going into the search circuit.



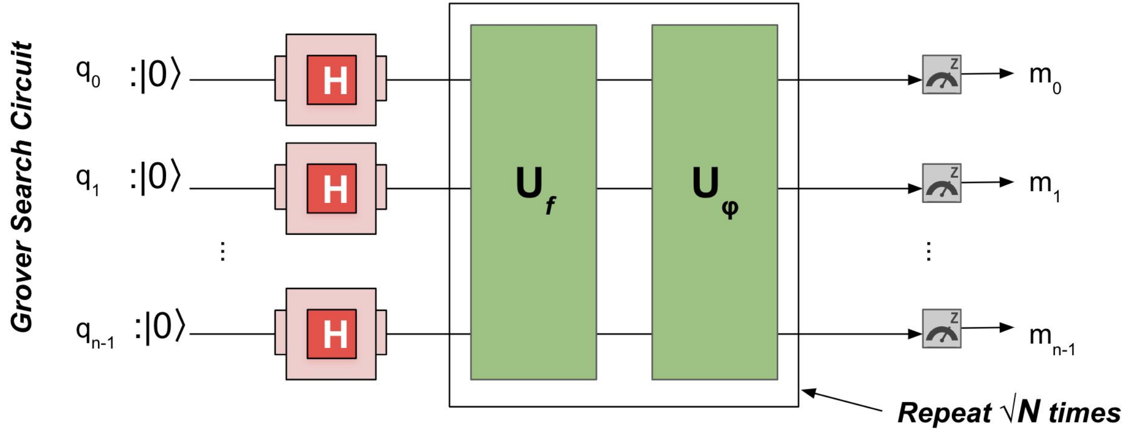
Grover's Search Mechanics

3. Construct an **Oracle, f** and a gate/matrix $\mathbf{U}_f$ that flips the sign of the object of search
4. Run $\mathbf{U}_f$ that flips the sign of the object of search followed by $\mathbf{U}_\varphi$ the circuit that inverts around the mean (we call this the **Grover Diffusion Operator**). Repeat this step **√N times**



Grover’s Search Mechanics

5. Measure and read off the state with an encoding that corresponds to the desired item in the database. Map the resulting state back to the database
6. If answer is incorrect, repeat steps 1-5. The chance of error is O(1/N)


Grover Search Circuit
q_0 :|0⟩
q_1 :|0⟩
q_{n-1} :|0⟩
H
H
H
U_f
U_φ
Z
m_0
m_1
m_{n-1}
Repeat √N times