CSCI 4140
Grover's Algorithm

Mark Green

Faculty of Science

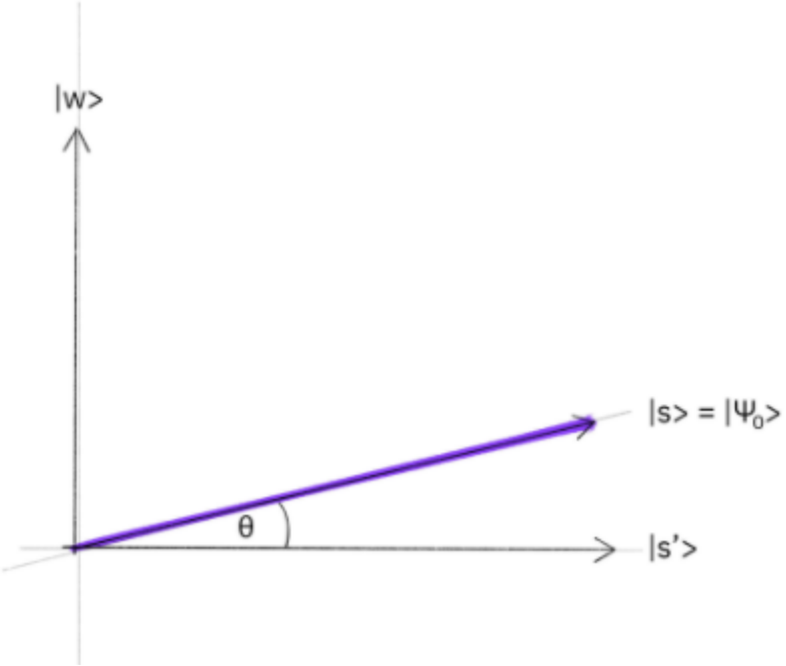
Ontario Tech

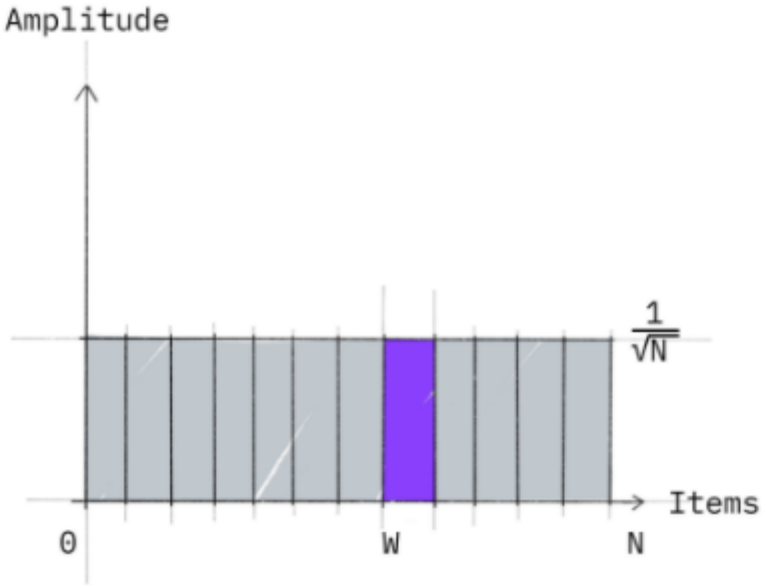
I love quantum computing

Introduction

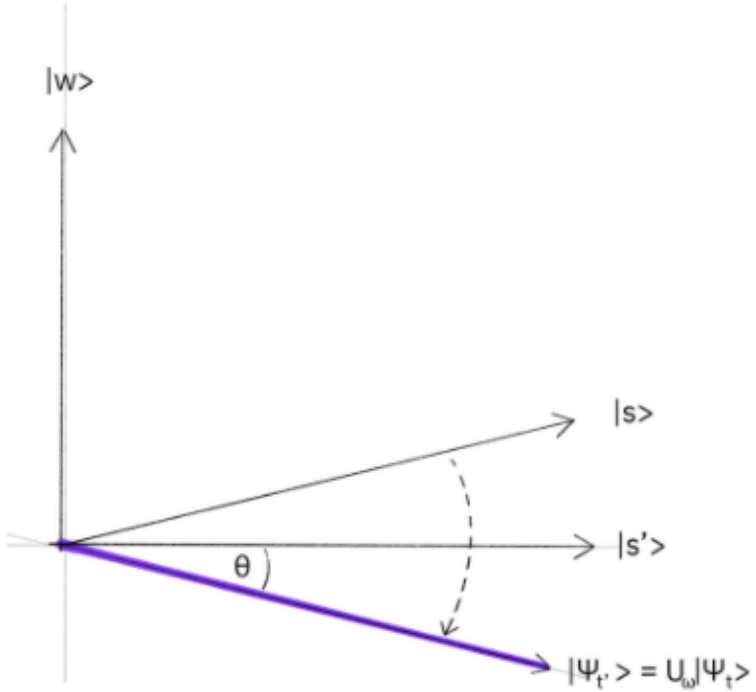
- In this video lecture we will investigate the implementation of Grover’s algorithm in Qiskit
- Recall that Grover’s algorithm is used for searching unstructured data for the location of an item w
- We construct an oracle for this problem that produces a +1 for locations that don’t have w, and -1 for locations that do have w
- We then use amplitude amplification to increase the probability of measuring w’s location

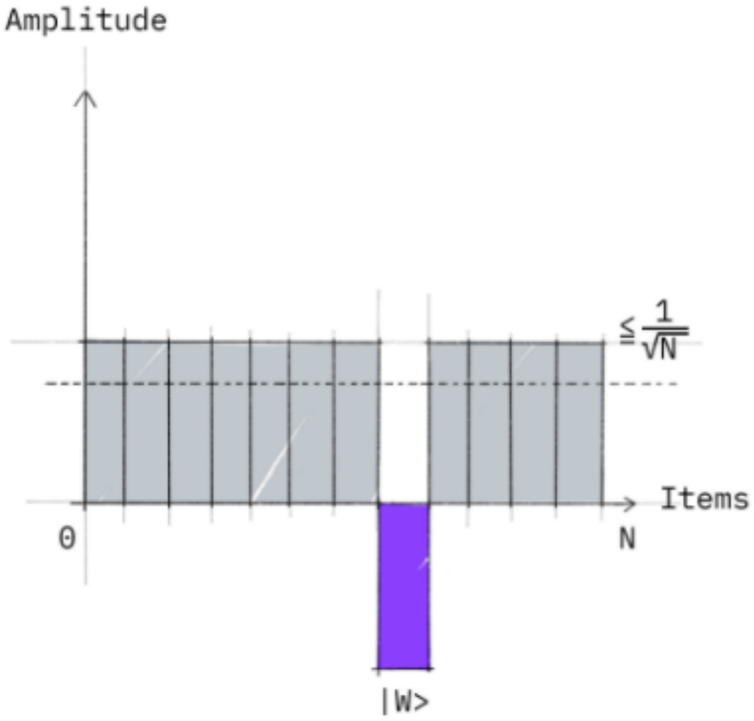
Introduction





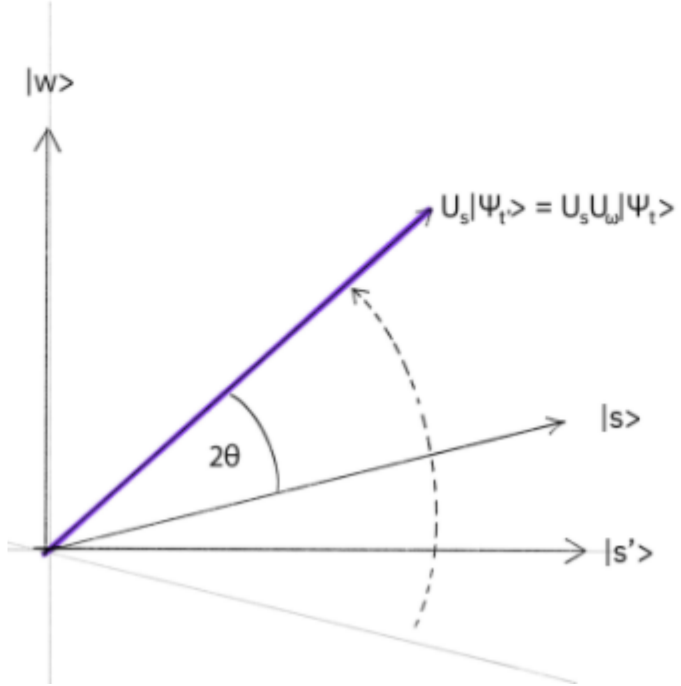
Introduction

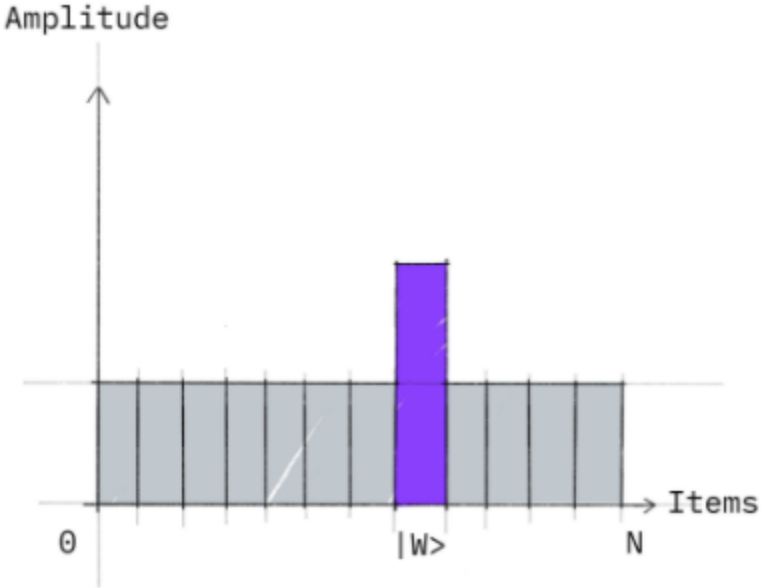




Introduction

Here $U_s = |s\rangle\langle s| - 1$
It's called the diffuser





Grover’s Algorithm

- If there are N items in the list, and w appears only once it takes $\sqrt{N}$ iterations of the algorithm at most to find it
- If the item we are looking for appears M times in the list it takes $\sqrt{N/_M}$ iterations of the algorithm
- We will start with a simple case of a list that has only 4 items, in this case we only need 2 qubits
- The item that we are looking is |11>

Grover’s Algorithm

- We want our oracle to perform the following operation:

$$U_\omega|s\rangle = U_\omega\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle).$$

- This can be done using the following matrix:

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

Note that this is a Controlled Z gate

Grover's Algorithm

- As usual start with the import statements:

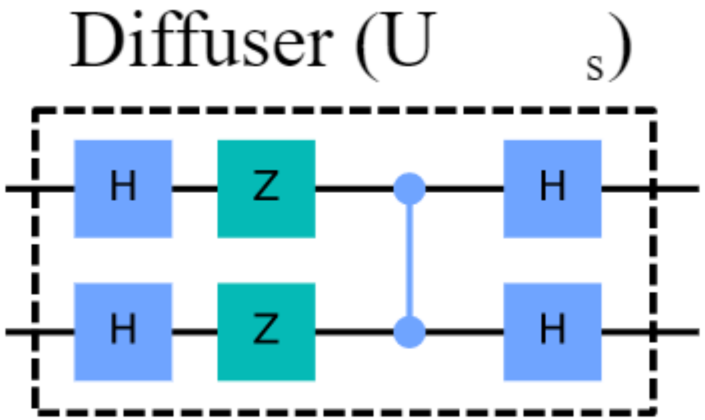
```
import matplotlib.pyplot as plt
import numpy as np
from qiskit import Aer, QuantumCircuit, ClassicalRegister, QuantumRegister, execute
from qiskit.quantum_info import Statevector
from qiskit.visualization import plot_histogram
```

- Also have a function for establishing the initial state:

```
def initialize_s(qc, qubits):
    """Apply a H-gate to 'qubits' in qc"""
    for q in qubits:
        qc.h(q)
    return qc
```

Grover's Algorithm

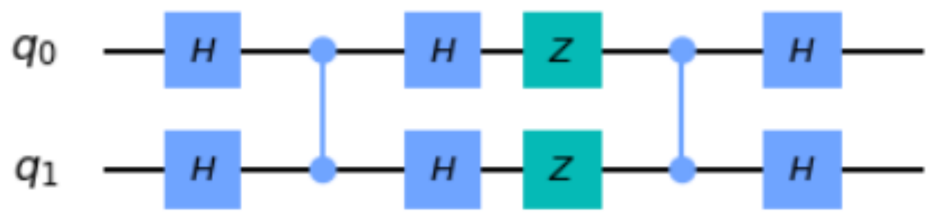
- In the case of two qubits we have the following diffuser circuit:



- Now we have all the pieces that we need, the code and the resulting circuit are shown on the next slide

Grover's Algorithm

```
n = 2
grover_circuit = QuantumCircuit(n)
grover_circuit = initialize_s(grover_circuit, [0,1])
grover_circuit.cz(0,1) # Oracle
# Diffusion operator (U_s)
grover_circuit.h([0,1])
grover_circuit.z([0,1])
grover_circuit.cz(0,1)
grover_circuit.h([0,1])
grover_circuit.draw('mpl')
```



Grover’s Algorithm

- If we use the state vector simulator we see that the circuit finds the location of |11>

```
sv_sim = Aer.get_backend('statevector_simulator')
job_sim = execute(grover_circuit, sv_sim)
statevec = job_sim.result().get_statevector()
from qiskit_textbook.tools import vector2latex
vector2latex(statevec, pretext="|\\psi\\rangle =")
```

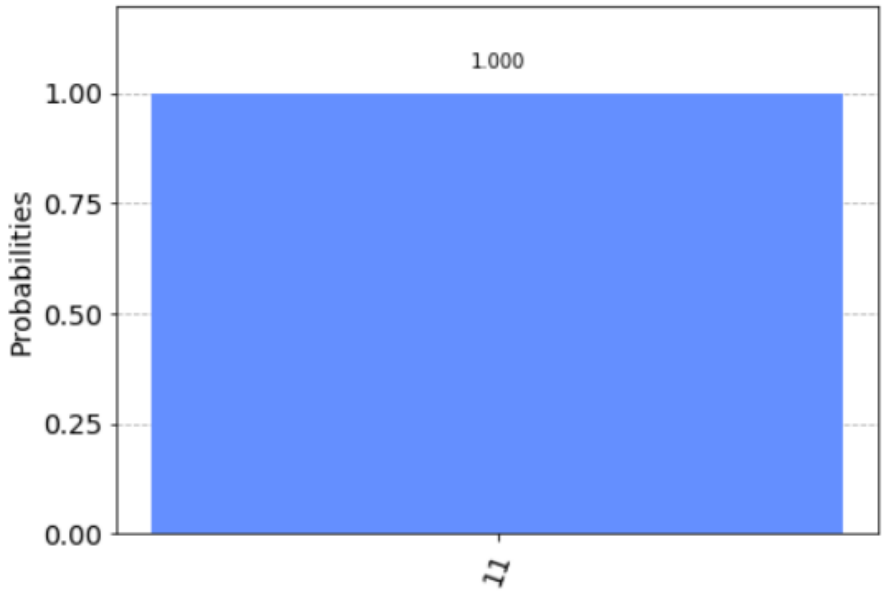
$$|\psi\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Grover’s Algorithm

- With the qasm simulator we get the following result:

```
grover_circuit.measure_all()

qasm_simulator = Aer.get_backend('qasm_simulator')
shots = 1024
results = execute(grover_circuit, backend=qasm_simulator, shots=shots).result()
answer = results.get_counts()
plot_histogram(answer)
```



Slightly More Complicated

- Examine a 3 qubit search with two values that we are looking for: |101> and |110>
- Our oracle now needs to produce the following:

$$|\psi_2\rangle = \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle - |101\rangle - |110\rangle + |111\rangle)$$

- This can be done with two controlled Z gates

Slightly More Complicated

- Our oracle can be constructed in the following way:

```
qc = QuantumCircuit(3)
qc.cz(0, 2)
qc.cz(1, 2)
oracle_ex3 = qc.to_gate()
oracle_ex3.name = "U$_\omega$"
```

- To make our life easier in the future, the Qiskit textbook provides a function that will construct a diffuser for an arbitrary number of bits
- This is shown on the next slide

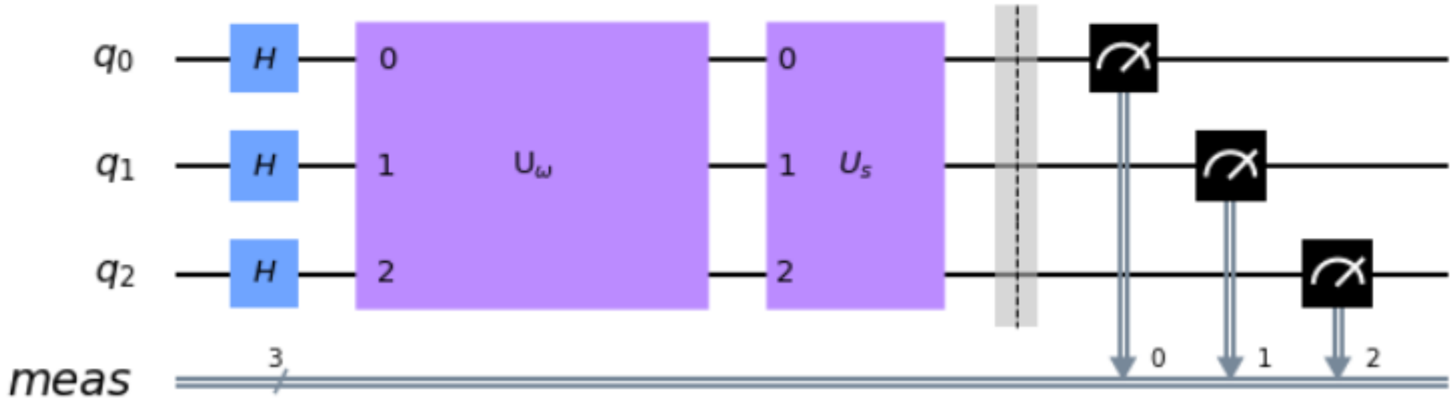
General Purpose Diffuser

```python
def diffuser(nqubits):
    qc = QuantumCircuit(nqubits)
    # Apply transformation |s> -> |00..0> (H-gates)
    for qubit in range(nqubits):
        qc.h(qubit)
    # Apply transformation |00..0> -> |11..1> (X-gates)
    for qubit in range(nqubits):
        qc.x(qubit)
    # Do multi-controlled-Z gate
    qc.h(nqubits-1)
    qc.mct(list(range(nqubits-1)), nqubits-1)  # multi-controlled-toffoli
    qc.h(nqubits-1)
    # Apply transformation |11..1> -> |00..0>
    for qubit in range(nqubits):
        qc.x(qubit)
    # Apply transformation |00..0> -> |s>
    for qubit in range(nqubits):
        qc.h(qubit)
    # We will return the diffuser as a gate
    U_s = qc.to_gate()
    U_s.name = "$U_s$"
    return U_s
```

Slightly More Complicated

- Putting this all together we get the following:

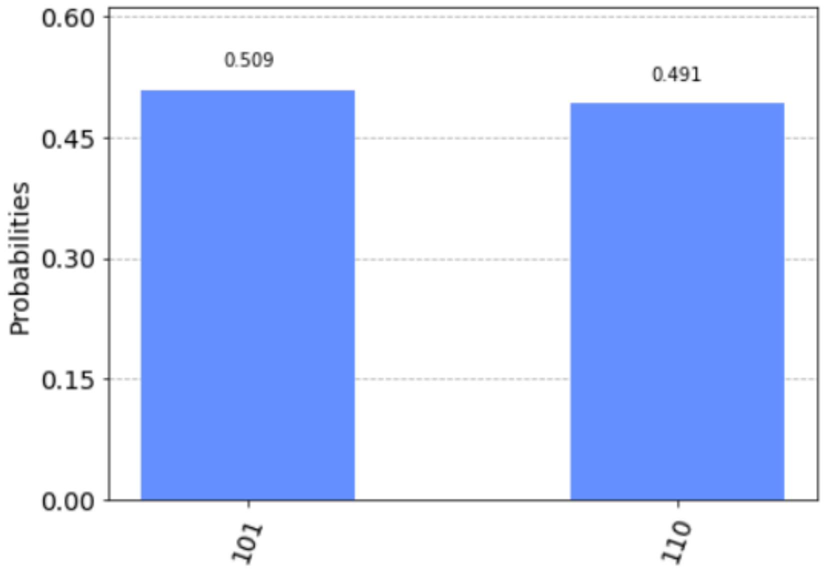
```
n = 3
grover_circuit = QuantumCircuit(n)
grover_circuit = initialize_s(grover_circuit, [0,1,2])
grover_circuit.append(oracle_ex3, [0,1,2])
grover_circuit.append(diffuser(n), [0,1,2])
grover_circuit.measure_all()
grover_circuit.draw('mpl')
```



Slightly More Complicated

- In this case we have N=3 and M=2, so we can solve this problem with a single iteration of our algorithm

```
backend = Aer.get_backend('qasm_simulator')
results = execute(grover_circuit, backend=backend, shots=1024).result()
answer = results.get_counts()
plot_histogram(answer)
```



Generalization

- In both examples we only needed one iteration, but what happens if we need more?
- With N bits in general we will need $\sqrt{N}$ iterations
- We can do this by repeating the oracle and the diffuser $\sqrt{N}$ times
- There is no way we can do a loop on a quantum computer, we need to repeat the gates, which increases the size of the circuit
- We will investigate this in the laboratory

Summary

- Investigated the basic techniques for implementing Grover’s algorithm
- The real problem is in the construction of the oracle
- Presented a procedure that will generate a diffuser for an arbitrary number of bits
- In general, the basic components of the circuit need to be repeated $\sqrt{N}$ times