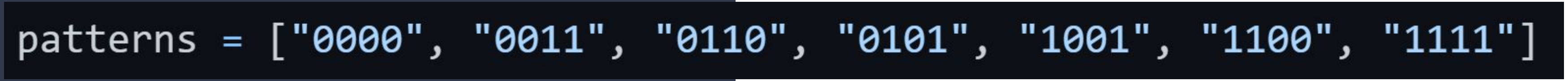
Quantum Associative Memory

Dan Ventura, Tony Martinez

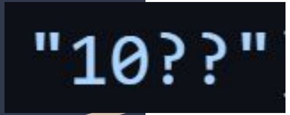
Aaron and Tarun

What is Associative Memory?

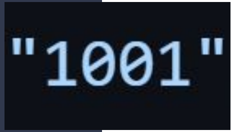
- To learn and remember the relationship between unrelated items
- Store memory and perform searches to determine the value given some query



```
patterns = ["0000", "0011", "0110", "0101", "1001", "1100", "1111"]
```



```
"10??"
```



```
"1001"
```

- If we store our patterns on qubits, can we then store 2^n bits?

Shoutouts

Dan Ventura and Tony Martinez

(QuAM.paper) (arxiv.org)

Saving Patterns

1. Convert x register to one of desired stored patterns and flip c1 if it's 0

2. Split our state into an uneven superposition: one "small" and one "large"

3. Save our state by applying a not gate on c1. States with a c register of 01 are now saved.

4. Repeat m times

$$\mathcal{P} = \{01, 10, 11\}$$

$$\boxed{|\overset{x}{00}, \overset{g}{0}, \overset{c}{00}\rangle \xrightarrow{FLIP} |01,0,10\rangle}$$

$$\boxed{\xrightarrow{\hat{S}^3} \frac{1}{\sqrt{3}}|01,0,11\rangle + \sqrt{\frac{2}{3}}|01,0,10\rangle}$$

$$\boxed{\xrightarrow{SAVE} \frac{1}{\sqrt{3}}|01,0,01\rangle + \sqrt{\frac{2}{3}}|01,0,00\rangle}$$

$$\xrightarrow{FLIP} \frac{1}{\sqrt{3}}|01,0,01\rangle + \sqrt{\frac{2}{3}}|10,0,10\rangle$$

$$\xrightarrow{\hat{S}^2} \frac{1}{\sqrt{3}}|01,0,01\rangle + \frac{1}{\sqrt{2}}\sqrt{\frac{2}{3}}|10,0,11\rangle + \sqrt{\frac{1}{2}}\sqrt{\frac{2}{3}}|10,0,10\rangle$$

$$\xrightarrow{SAVE} \frac{1}{\sqrt{3}}|01,0,01\rangle + \frac{1}{\sqrt{3}}|10,0,01\rangle + \sqrt{\frac{1}{3}}|10,0,00\rangle$$

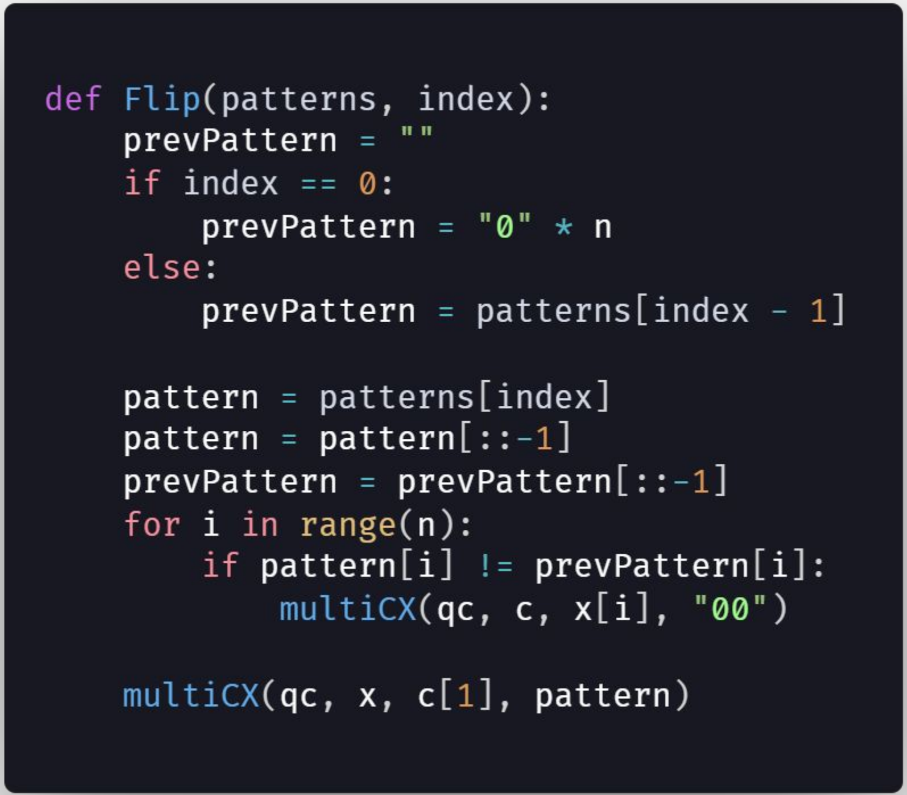
$$\xrightarrow{FLIP} \frac{1}{\sqrt{3}}|01,0,01\rangle + \frac{1}{\sqrt{3}}|10,0,01\rangle + \sqrt{\frac{1}{3}}|11,0,10\rangle$$

$$\xrightarrow{\hat{S}^1} \frac{1}{\sqrt{3}}|01,0,01\rangle + \frac{1}{\sqrt{3}}|10,0,01\rangle + \frac{1}{\sqrt{1}}\sqrt{\frac{1}{3}}|11,0,11\rangle + \sqrt{\frac{0}{1}}\sqrt{\frac{1}{3}}|11,0,10\rangle$$

$$\xrightarrow{SAVE} \frac{1}{\sqrt{3}}|01,0,01\rangle + \frac{1}{\sqrt{3}}|10,0,01\rangle + \frac{1}{\sqrt{3}}|11,0,01\rangle$$

$$-\frac{1}{\sqrt{3}}|01\rangle + \frac{1}{\sqrt{3}}|10\rangle - \frac{1}{\sqrt{3}}|11\rangle,$$

Flip

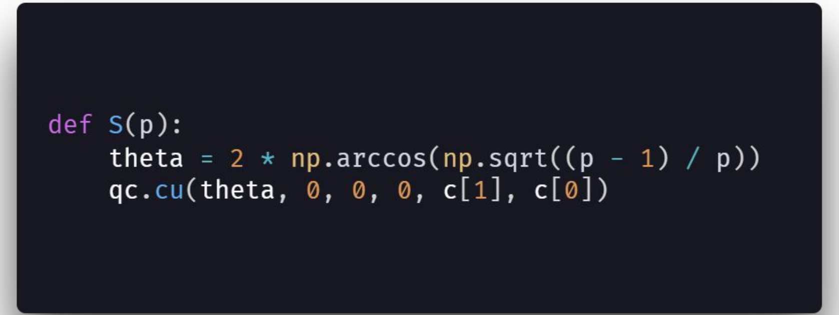


```python
def Flip(patterns, index):
    prevPattern = ""
    if index == 0:
        prevPattern = "0" * n
    else:
        prevPattern = patterns[index - 1]

    pattern = patterns[index]
    pattern = pattern[::-1]
    prevPattern = prevPattern[::-1]
    for i in range(n):
        if pattern[i] != prevPattern[i]:
            multiCX(qc, c, x[i], "00")

    multiCX(qc, x, c[1], pattern)
```

S Operator



```python
def S(p):
    theta = 2 * np.arccos(np.sqrt((p - 1) / p))
    qc.cu(theta, 0, 0, 0, c[1], c[0])
```

$$\begin{bmatrix} \sqrt{\frac{p-1}{p}} & -\frac{1}{\sqrt{p}} \\ \frac{1}{\sqrt{p}} & \sqrt{\frac{p-1}{p}} \end{bmatrix}$$

S Operator Examples

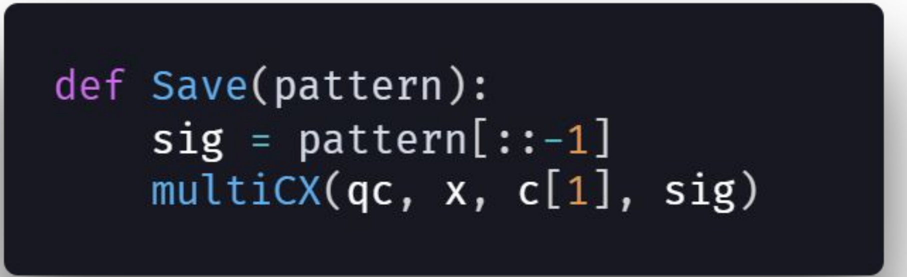
p = 2

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

p = 3

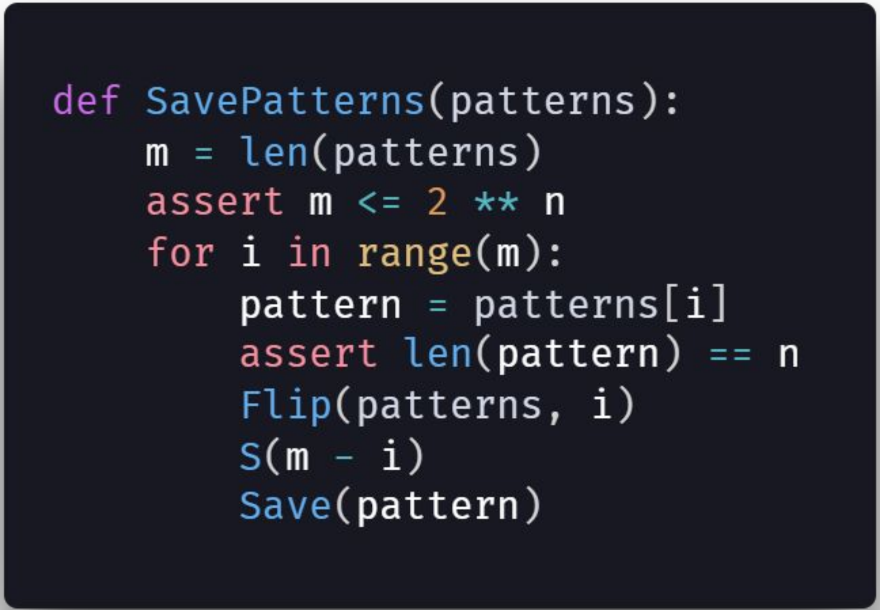
$$\begin{bmatrix} \frac{2}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{3}} \end{bmatrix}$$

Save



```python
def Save(pattern):
    sig = pattern[::-1]
    multiCX(qc, x, c[1], sig)
```

Saving Patterns



```python
def SavePatterns(patterns):
    m = len(patterns)
    assert m <= 2 ** n
    for i in range(m):
        pattern = patterns[i]
        assert len(pattern) == n
        Flip(patterns, i)
        S(m - i)
        Save(pattern)
```

Searching Patterns

Grover's refresher

$$R = \frac{\pi}{4}\sqrt{\frac{N}{M}}$$

$|0000\rangle$ $|1111\rangle$

$$|\psi\rangle = (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

1. Hadamard everything

$$|\psi\rangle \xrightarrow{\hat{W}} |\psi\rangle = \frac{1}{4}(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1).$$

2. Invert the phase of desired state (phase kickback) $|0110\rangle$

$$|\psi\rangle \xrightarrow{\hat{I}_\tau} |\psi\rangle = \frac{1}{4}(1,1,1,1,1,1,-1,1,1,1,1,1,1,1,1,1)$$

3. Rotate states about average (W operator)

$$|\psi\rangle \xrightarrow{\hat{G}} |\psi\rangle = \frac{1}{16}(3,3,3,3,3,3,11,3,3,3,3,3,3,3,3,3).$$

$$|\psi\rangle \xrightarrow{\hat{I}_\tau} |\psi\rangle = \frac{1}{16}(3,3,3,3,3,3,-11,3,3,3,3,3,3,3,3,3)$$

$$|\psi\rangle \xrightarrow{\hat{G}} |\psi\rangle = \frac{1}{64}(5,5,5,5,5,5,61,5,5,5,5,5,5,5,5,5)$$

$$|\psi\rangle \xrightarrow{\hat{I}_\tau} |\psi\rangle = \frac{1}{64}(5,5,5,5,5,5,-61,5,5,5,5,5,5,5,5,5).$$

$$|\psi\rangle \xrightarrow{\hat{G}} |\psi\rangle = \frac{1}{256}(-13,-13,-13,-13,-13,-13,251,-13,-13,-13,-13,-13,-13,-13,-13,-13)$$

$$\frac{251}{256} \approx 0.96$$

Grover's?

- No longer start with all possible states

$$|\psi\rangle = \frac{1}{\sqrt{6}}(1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1).$$

$$|\psi\rangle \xrightarrow{\hat{I}_\tau} |\psi\rangle = \frac{1}{\sqrt{6}}(1,0,0,1,0,0,-1,0,0,1,0,0,1,0,0,1).$$

$$|\psi\rangle \xrightarrow{\hat{G}} |\psi\rangle = \frac{1}{2\sqrt{6}}(-1,1,1,-1,1,1,3,1,1,-1,1,1,-1,1,1,-1).$$

$$|\psi\rangle \xrightarrow{\hat{I}_\tau} |\psi\rangle = \frac{1}{2\sqrt{6}}(-1,1,1,-1,1,1,-3,1,1,-1,1,1,-1,1,1,-1).$$

$$|\psi\rangle \xrightarrow{\hat{G}} |\psi\rangle = \frac{1}{8\sqrt{6}}(5,-3,-3,5,-3,-3,13,-3,-3,5,-3,-3,5,-3,-3,5).$$

$$\frac{13}{8\sqrt{6}} \approx 0.66$$

Modified Grover's

- Add rotation operator that rotates all stored states

$$|\psi\rangle = \frac{1}{\sqrt{6}}(1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1).$$

$$|\psi\rangle \xrightarrow{\hat{I}_\tau} |\psi\rangle = \frac{1}{\sqrt{6}}(1,0,0,1,0,0,-1,0,0,1,0,0,1,0,0,1).$$

$$|\psi\rangle \xrightarrow{\hat{G}} |\psi\rangle = \frac{1}{2\sqrt{6}}(-1,1,1,-1,1,1,3,1,1,-1,1,1,-1,1,1,-1)$$

$$|\psi\rangle \xrightarrow{\hat{I}_p} |\psi\rangle = \frac{1}{2\sqrt{6}}(1,1,1,1,1,1,-3,1,1,1,1,1,1,1,1,1)$$

$$|\psi\rangle \xrightarrow{\hat{G}} |\psi\rangle = \frac{1}{4\sqrt{6}}(1,1,1,1,1,1,9,1,1,1,1,1,1,1,1,1).$$

$$|\psi\rangle \xrightarrow{\hat{I}_\tau} |\psi\rangle = \frac{1}{4\sqrt{6}}(1,1,1,1,1,1,-9,1,1,1,1,1,1,1,1,1)$$

$$|\psi\rangle \xrightarrow{\hat{G}} |\psi\rangle = \frac{1}{16\sqrt{6}}(-1,-1,-1,-1,-1,-1,39,-1,-1,-1,-1,-1,-1,-1,-1,-1)$$

$$R - 2 = \frac{\pi}{4}\sqrt{\frac{N}{M}} - 2$$

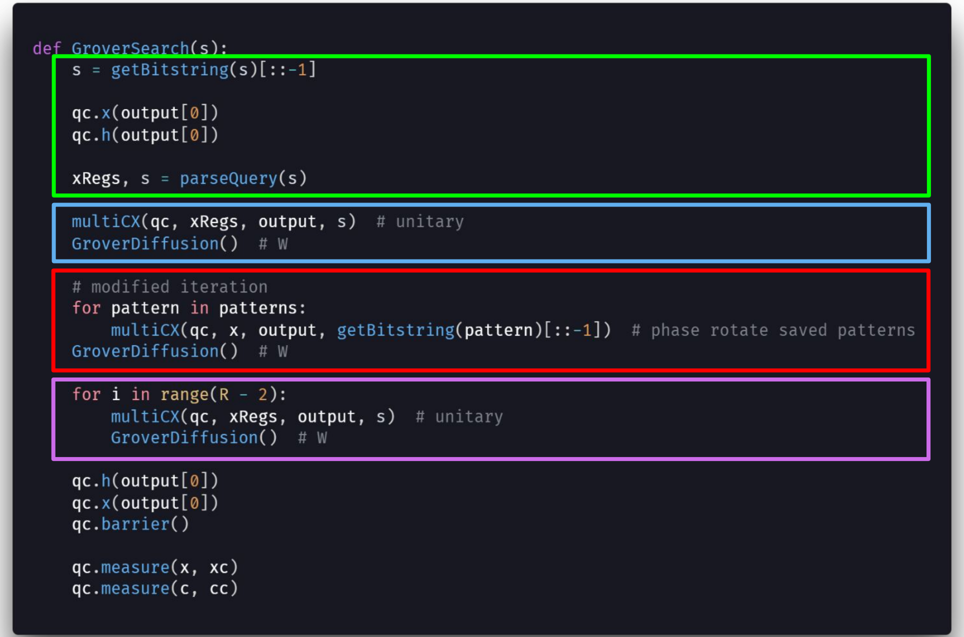
$$\frac{39}{16\sqrt{6}} \approx 0.995$$

$$\hat{I}_p$$

```
# modified iteration
for pattern in patterns:
    multiCX(qc, x, output, getBitstring(pattern)[::-1])  # phase rotate saved patterns
```

- This implementation classically stores the patterns, which defeats the purpose of the memory
- Doing this non-classically is difficult, we tried using auxiliary qubits, Quantum phase estimation, and QFT, but with no luck

Modified Grover's Implementation



```python
def GroverSearch(s):
    s = getBitstring(s)[::-1]

    qc.x(output[0])
    qc.h(output[0])

    xRegs, s = parseQuery(s)

    multiCX(qc, xRegs, output, s)  # unitary
    GroverDiffusion()  # W

    # modified iteration
    for pattern in patterns:
        multiCX(qc, x, output, getBitstring(pattern)[::-1])  # phase rotate saved patterns
    GroverDiffusion()  # W

    for i in range(R - 2):
        multiCX(qc, xRegs, output, s)  # unitary
        GroverDiffusion()  # W

    qc.h(output[0])
    qc.x(output[0])
    qc.barrier()

    qc.measure(x, xc)
    qc.measure(c, cc)
```

QuAM

Recap

1. Save patterns of bitstrings into a superposition of states
    1. Flip the qubits of the x register to match the bitstring we want to save
    2. Using the bitstring as a control, change the c register to mark the saved state as saved
    3. Split the state unevenly, using the small state to permanently save the state and keeping the larger one to save the rest of the bitstrings
    4. Save the state by setting the c register of the permanent state to 01
2. Run a modified Grover’s algorithm with a given bitstring as input to amplify that state and measure it
    1. Run the first iteration of a normal Grover’s algorithm
    2. Apply a phase rotation to the states that were originally saved in step 1
    3. Apply the Grover Diffusion operator
    4. Resume the normal Grover’s algorithm for R – 2 iterations

Demo

Results/Drawbacks

$$R = \frac{\pi}{4}\sqrt{\frac{N}{M}}$$

- Bad at searching for multi-solution queries
  - Ex. We saved 0011 and 0000 and we search for 00??
  - {'01 1011': 57, '01 0010': 29, '01 0101': 20, '01 0000': 251, '01 0011': 264, '01 0111': 46, '01 1010': 55, '01 0100': 66, '01 1101': 48, '01 1110': 50, '01 1000': 53, '01 0110': 18, '01 1100': 18, '01 0001': 13, '01 1111': 18, '01 1001': 18}
  - Due to unknown M for calculating R iterations

- Applying Grover's algorithm and measuring destroys the superposition, so just accessing the memory once destroys it
  - This can be solved using controlled NOTs onto another register of equal size, similar to how we did 3-qubit repetition encoding. This takes time, but it will copy the state each time to keep reusing

Applications

$|x, y, g, c\rangle$

- Quantum Array
  - *x* register as key/index
  - *y* register as value
  - Save both registers
  - Apply Grover's only on *x*
  - Measure *y*
- Quantum Hashmap/Dictionary
  - Extension of an array
  - Using classical hashing functions and modular arithmetic to translate any value for the input into one that will fit into the *x* register
- Efficient storage for any bitstring
  - Extends the idea of an array to a bitmap
  - Can store a 64-bit integer with 7 qubits
    - 6 qubits store the index of the value
    - 7th qubit stores the value of that bit
  - Comes at a time cost, need to measure each value individually to reconstruct the full bitstring

Citations

D. Ventura, T. Martinez, Quantum associative memory with exponential capacity, in 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227) (IEEE, 1998)