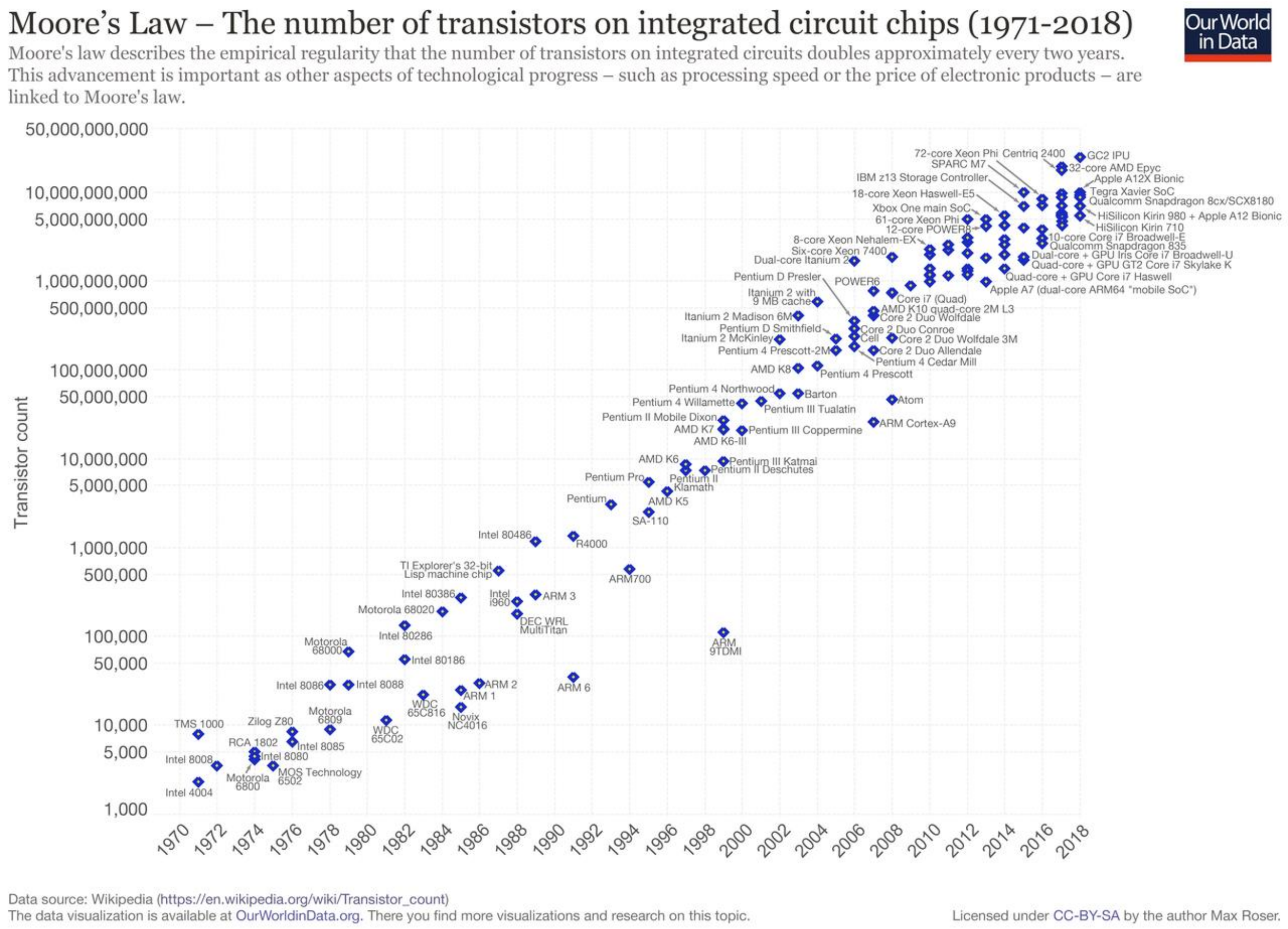
QUANTUM COMPUTATION AND SEARCH ALGORITHMS

A Historical Overview and Introduction

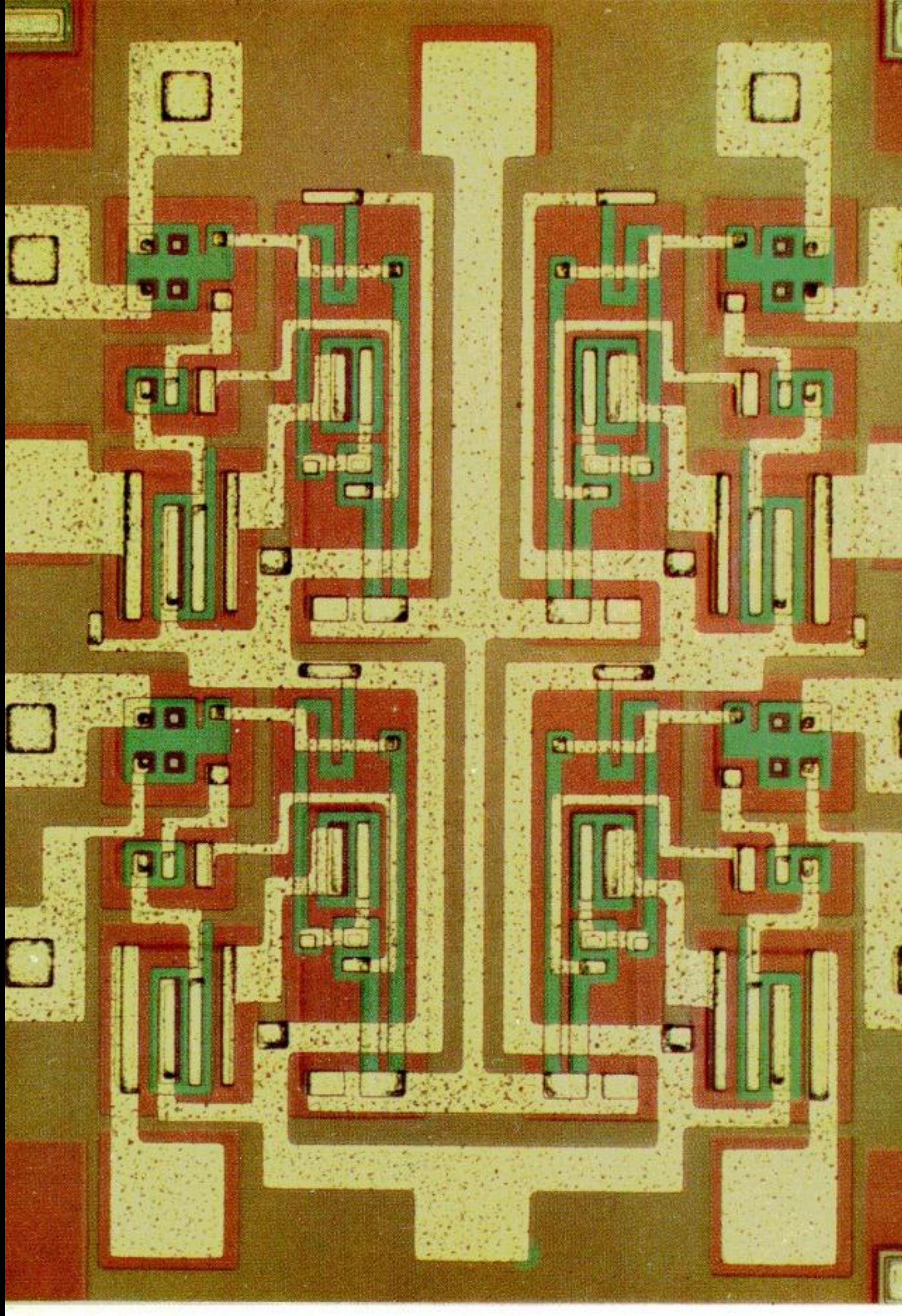
HISTORY

- Current computation based on silicon microchip
- Moore’s Law (1965) anticipates exponential growth of processing power
- In fact, Moore’s Law has held for half a century



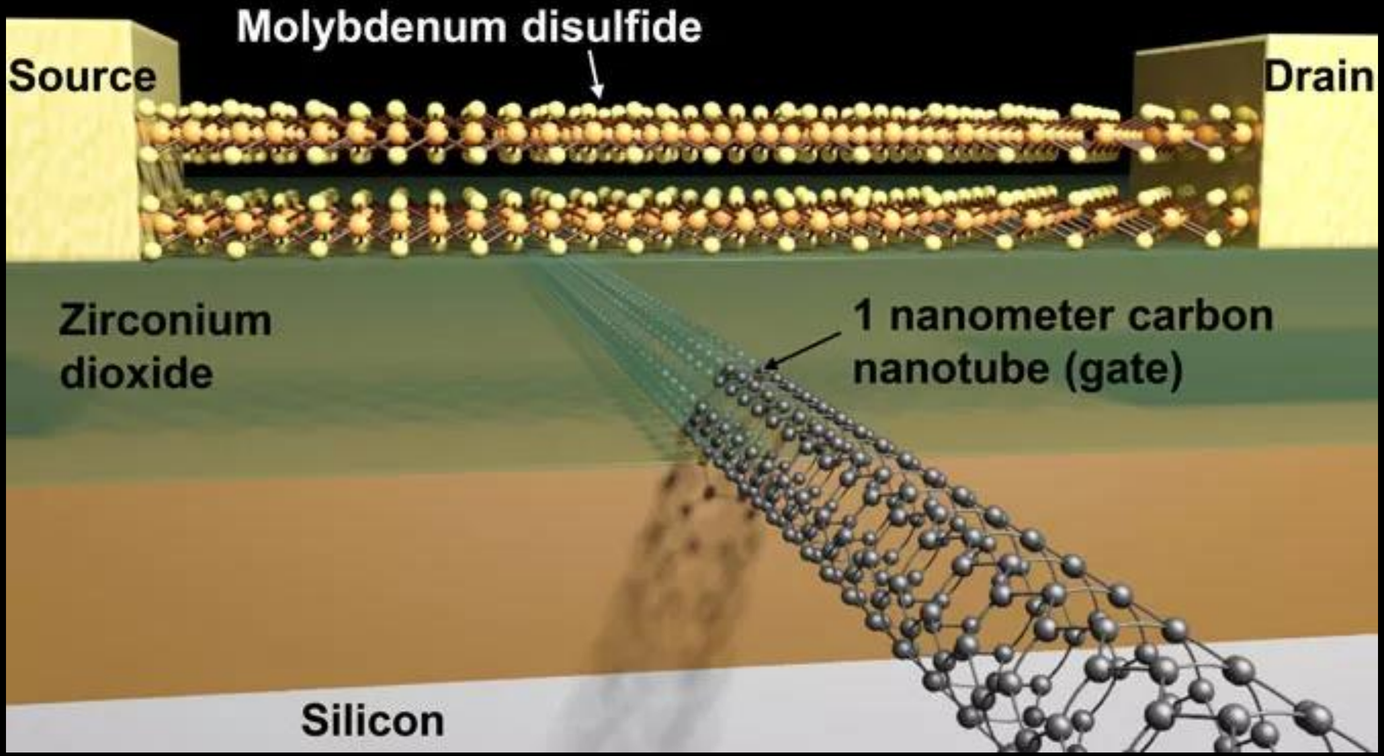
HISTORY

- Moore’s Law has limits, and must eventually fail
- So far, innovation has allowed the creation of ever smaller processors
- Currently, smallest is 5 nm (At right, 20 nm)



HISTORY

- As size decreases, physical laws place boundary on Moore's law
- Quantum transport techniques still currently extending the lifetime of Moore's Law
- Eventually atomic scale is hard limit on Moore's Law, as quantum mechanical effects predominate



HISTORY

- Problems classified according to computational complexity
- P if soluble in Polynomial time; NP if not.
- Conventional computers can handle P class problems
- Conventional computers struggle with NP class problems
- Integer factorization problem a classic NP problem
- NP problems may be intractable even for best supercomputers

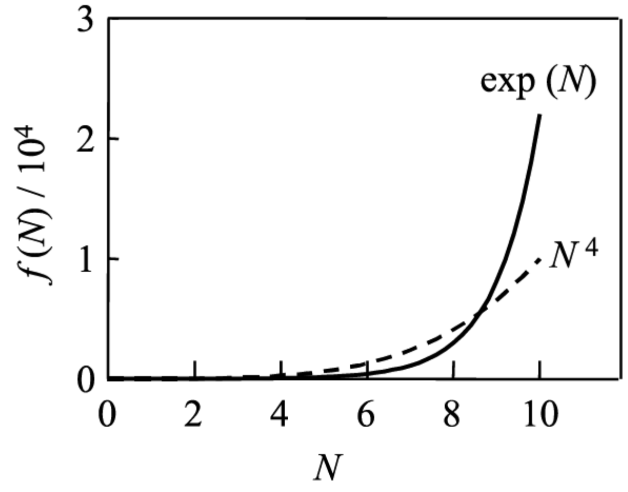
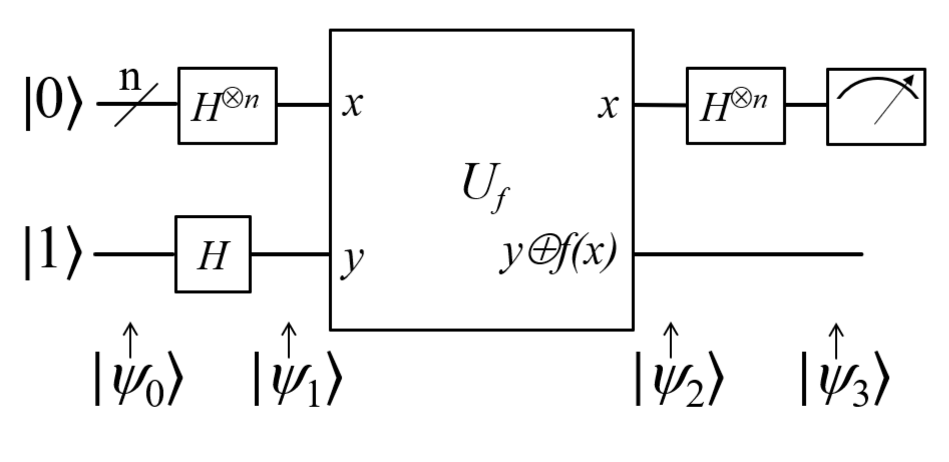


Fig. 4 Comparison of the size scaling of a polynomial function (N^4) with a non-polynomial function, namely $\exp(N)$.

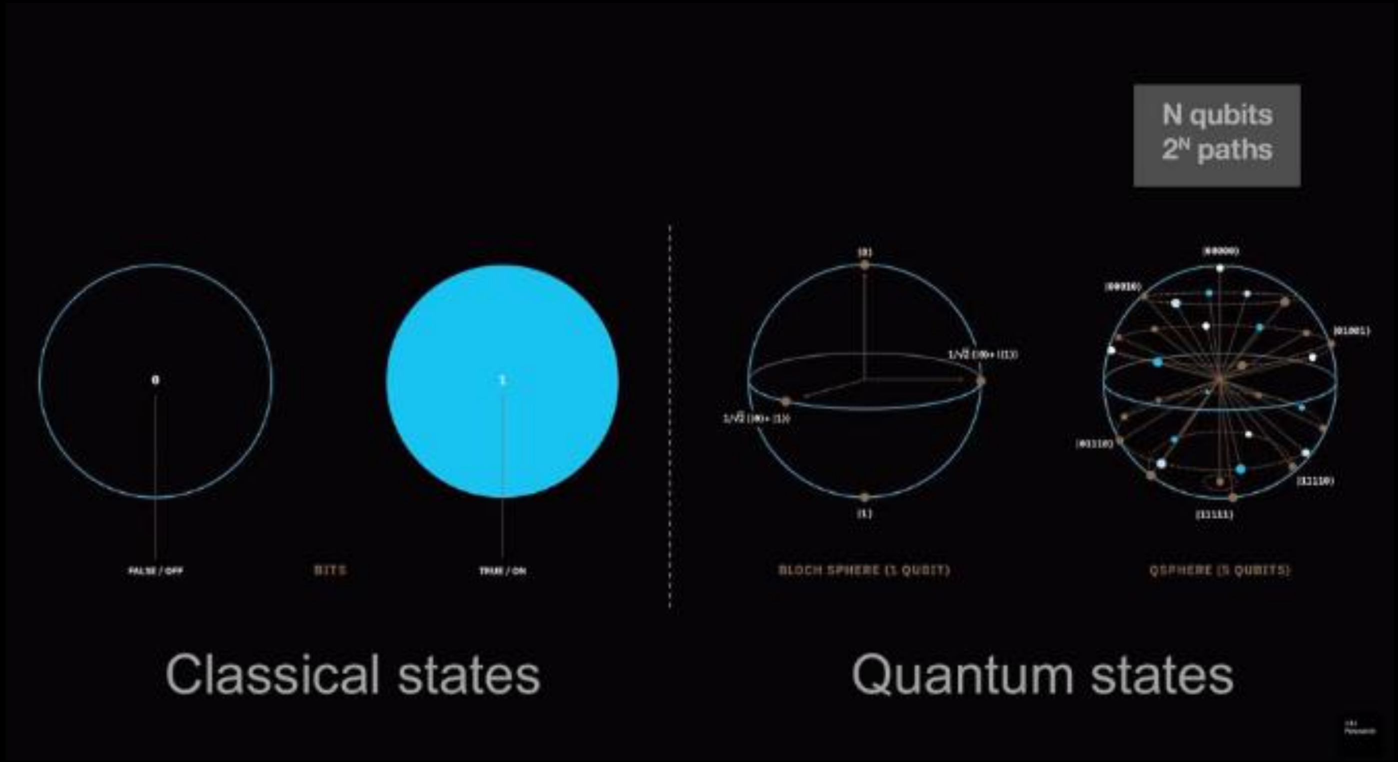
HISTORY



- Feynmann proposes “quantum computers” in 1982
- In 1985, David Deutsch identifies basic principles of quantum computation
- With Richard Jozsa, formulates Deutsch-Jozsa algorithm in 1992

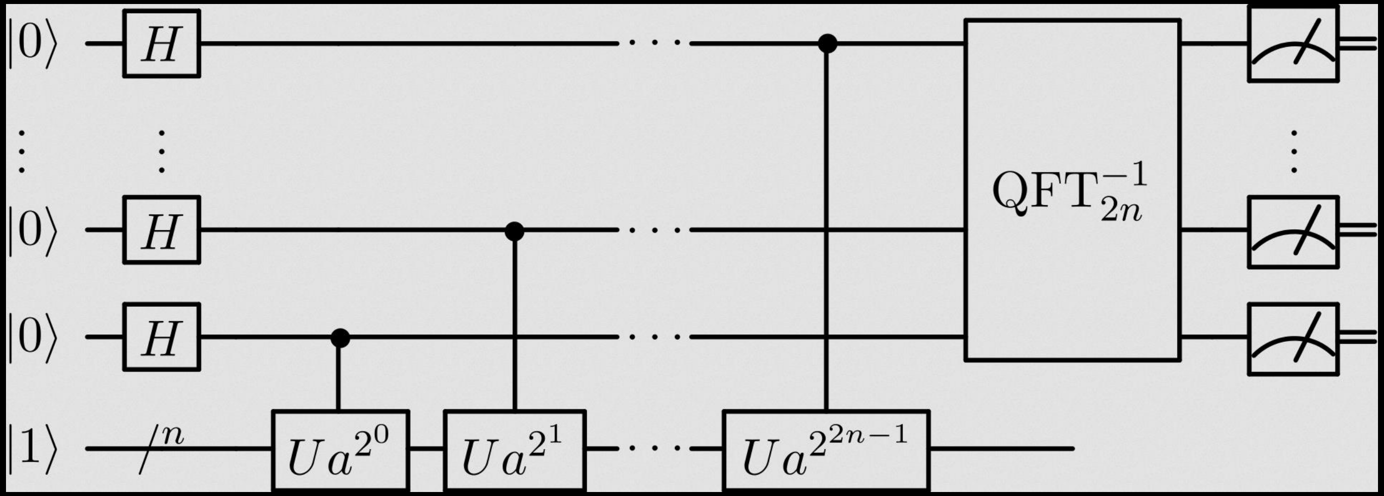
HISTORY

Information encoded as quantum states– superpositions of eigenstates



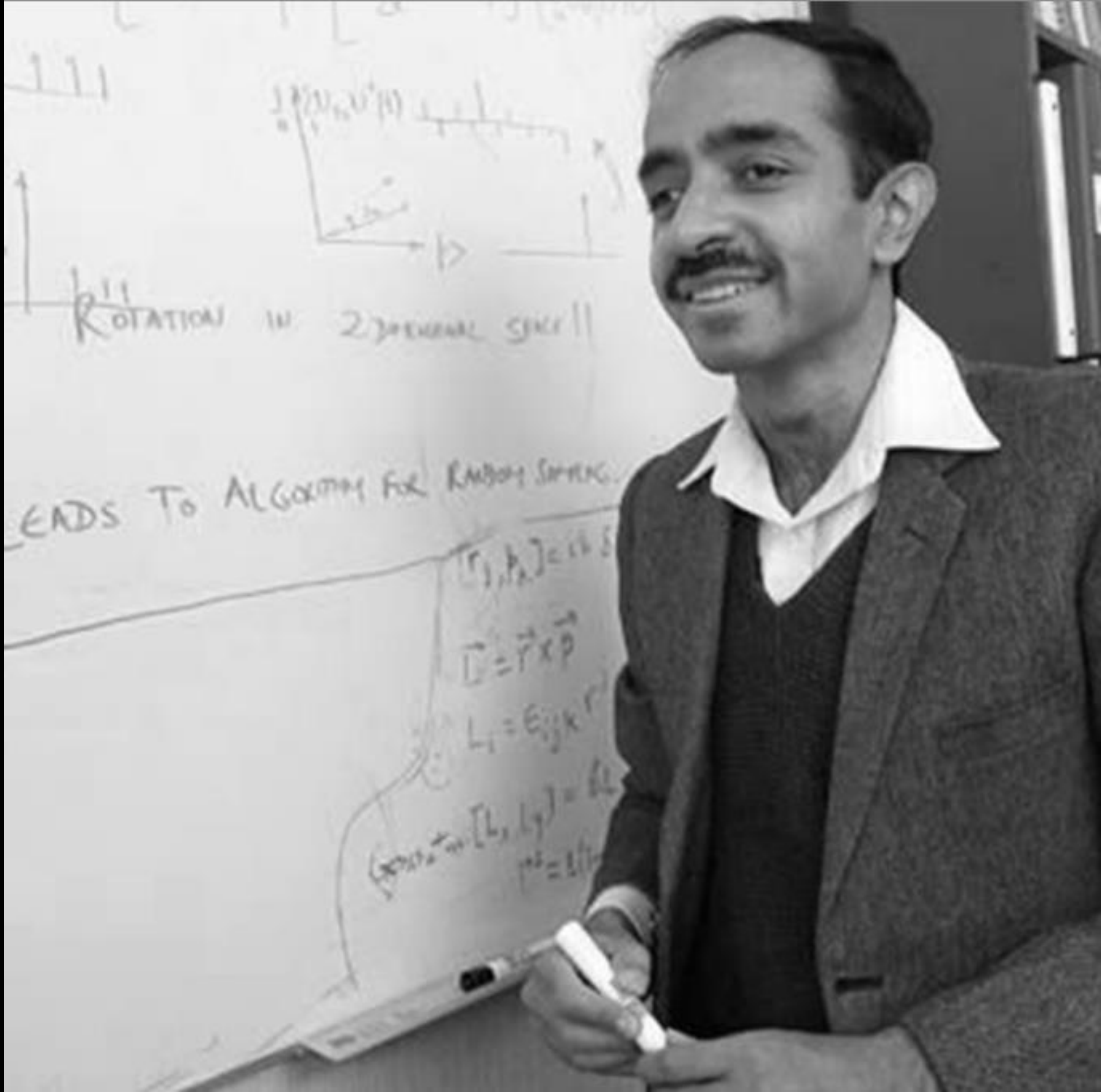
HISTORY

- 1994, Peter Shor develops his algorithm for the factorization problem
- Demonstrates that for a quantum computer, the factorization problem is of P class complexity



GROVER'S ALGORITHM

- 1996, Lov Grover described the algorithm that now bears his name
- Described as a means of searching and unsorted database
- For n possible values that need to be searched for a single correct value, is more efficient than similar conventional computer algorithms



Algorithmic Searching

Given an algorithm F,
find a number t
such that $F(t)=-1$

Searching for a unique target value t

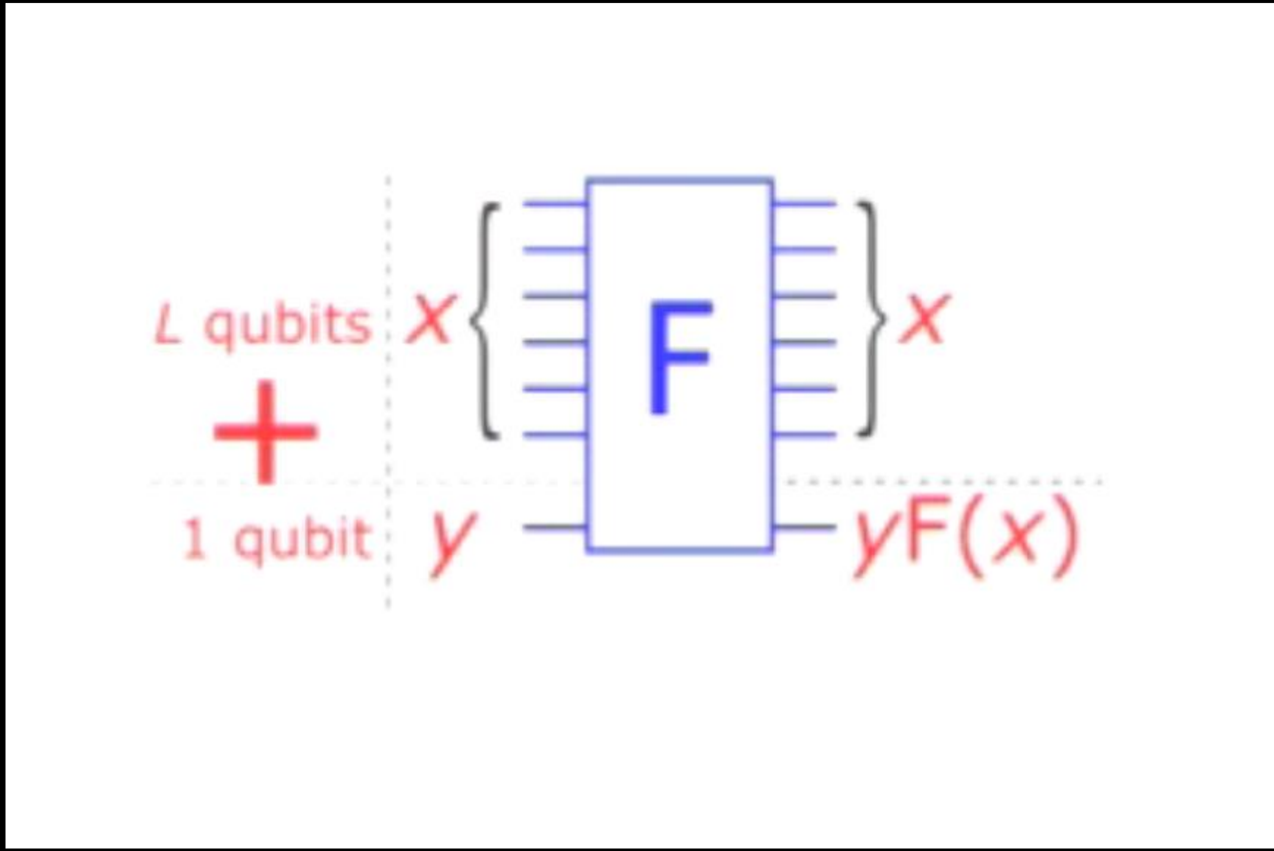
$$F(t)=-1$$
$$F(x)=1 \quad (\forall x \neq t)$$

OVERVIEW

- Performs an exhaustive search of N values for M desired values
- Tags correct value as -1, and wrongs values as 1
- The algorithm is a function which checks whether inputs are valid or not based on some criteria

OVERVIEW

- We call our algorithm, or function, an “Oracle”
- Provide it an L-qubit input, and one auxiliary qubit, to produce L+1 outputs
- How many trips to see the Oracle are needed to find our desired value?
- Answer: Classically, N-1
- Grover’s algorithm answer: You might be surprised!



FIRST SUBROUTINE

- Three subroutines comprise the algorithm
- First consists of a Hadamard gate
- Basically, places the initial "blank" ket into the equal superposition state

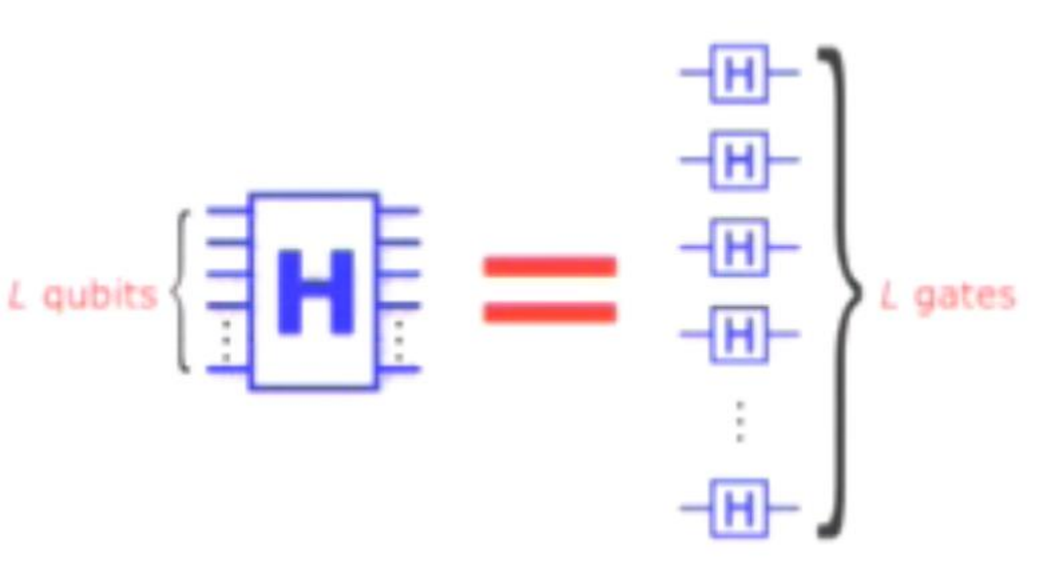
H Hadamard gates **H** operating on L qubits

States:

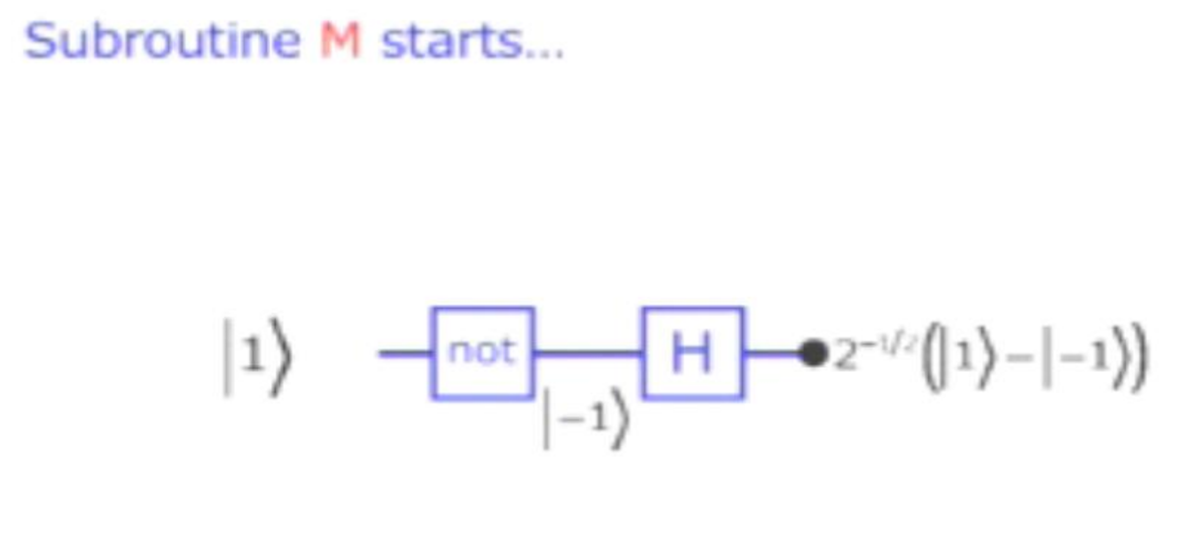
$$|\mu\rangle \equiv \mathbf{H}|\mathbf{0}\rangle = 2^{-L/2}\left(|1\rangle + |-1\rangle\right)\left(|1\rangle + |-1\rangle\right)\ldots$$
$$= 2^{-L/2}\sum_{x=0}^{N-1}|x\rangle$$

$$|\mathbf{0}\rangle \equiv |\overbrace{1,1,\ldots}^{L\text{ qubits}}\rangle \qquad \Rightarrow \langle x|\mu\rangle = 2^{-L/2} = N^{-1/2}$$

$$\mathbf{H}|\mu\rangle = |\mathbf{0}\rangle$$



SECOND SUBROUTINE



$$2^{-1/2}|x\rangle(|1\rangle - |-1\rangle) = 2^{-1/2}(|x,1\rangle - |x,-1\rangle)$$

$$\rightarrow 2^{-1/2}(|x,F(x)\rangle - |x,-F(x)\rangle)$$

$$= 2^{-1/2}|x\rangle(|F(x)\rangle - |-F(x)\rangle)$$

$$= 2^{-1/2}F(x)|x\rangle(|1\rangle - |-1\rangle)$$

$$M|x\rangle = F(x)|x\rangle$$

$$\Rightarrow \quad M = I - 2|t\rangle\langle t|$$

$$M|t\rangle = -|t\rangle$$

$$M|x\rangle = |x\rangle \quad (\forall x \neq t)$$

$$M|t\rangle = -|t\rangle$$

$$M|x\rangle = |x\rangle \quad (\forall x \neq t)$$

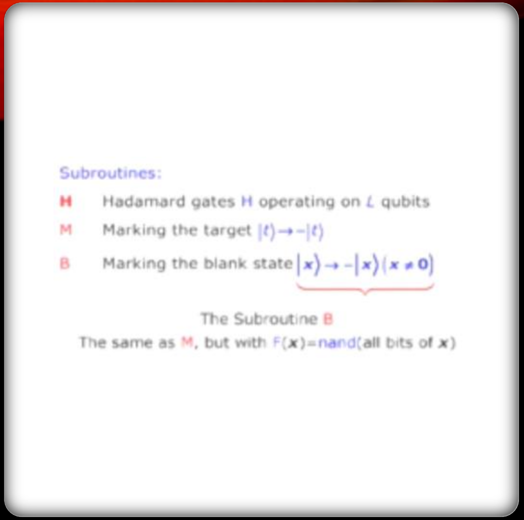
$$M|\mu\rangle = N^{-1/2} M \sum_{x=0}^{N-1} |x\rangle$$

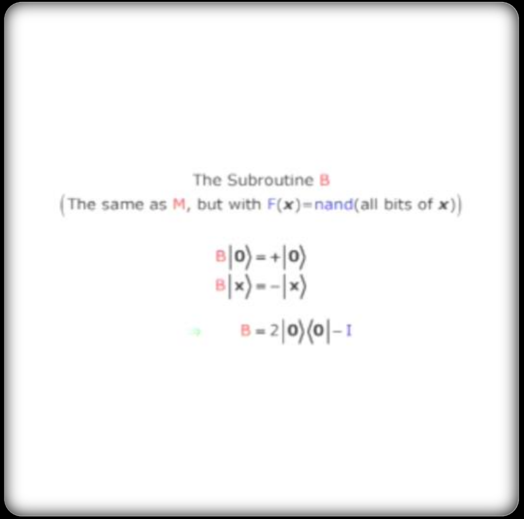
$$= N^{-1/2} \sum_{x=0}^{N-1} F(x)|x\rangle$$

$$\Rightarrow M|\mu\rangle = |\mu\rangle - 2N^{-1/2}|t\rangle$$

- The Marking Subroutine
- "Marks" the correct input -1, and leaves undesired inputs unchanged

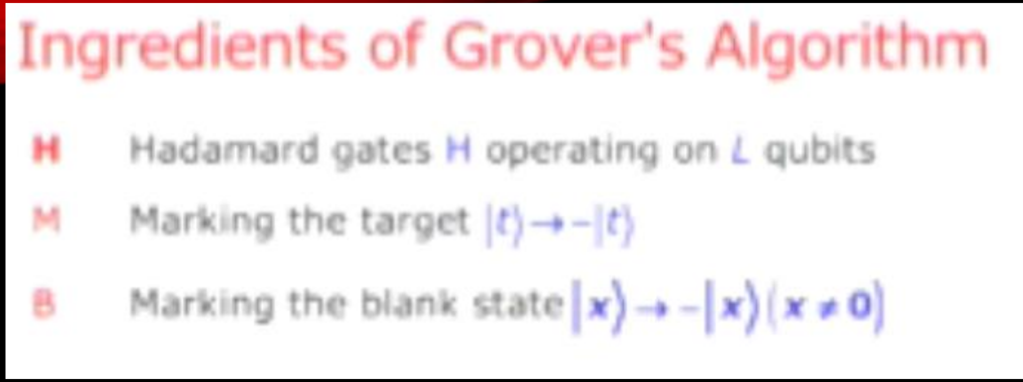
THIRD SUBROUTINE

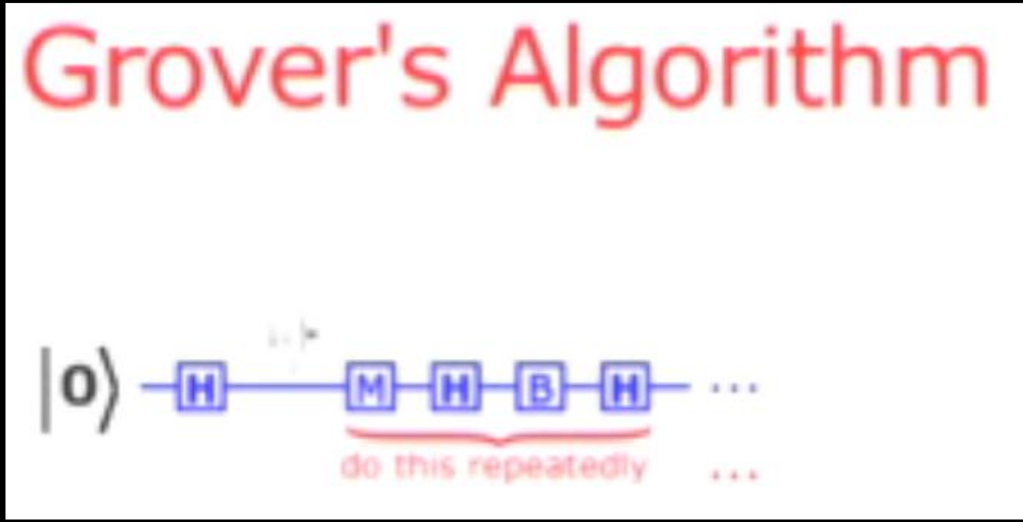


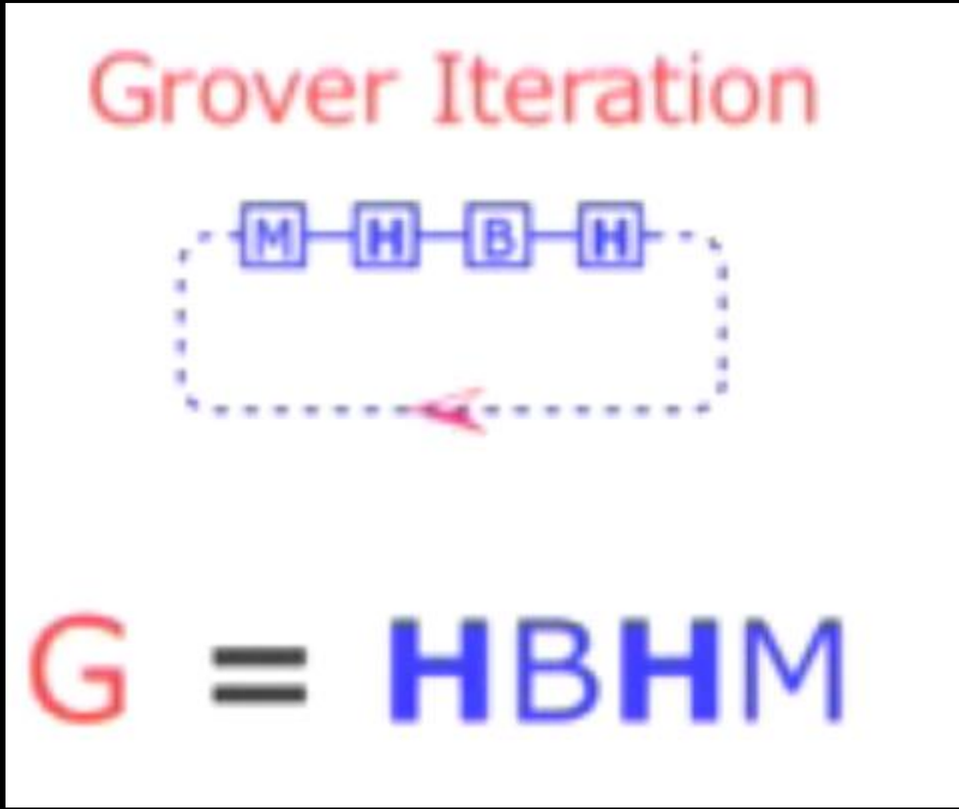


- Tags all kets except the zero ket with -1

THE FULL ALGORITHM



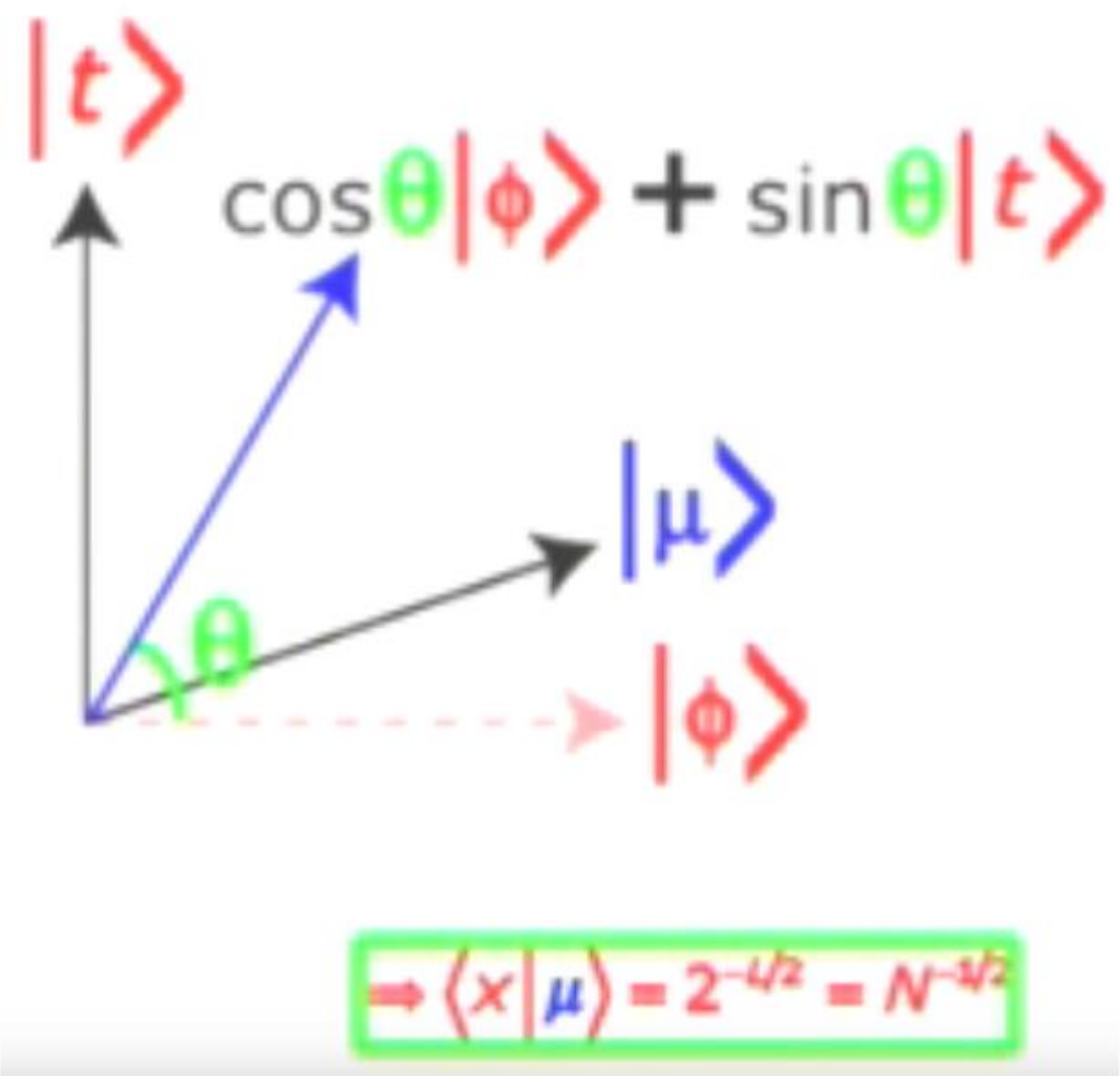




- Combine the three subroutines to produce G = HBHM
- G the "Grover Iteration", the unitary operation which perform the search

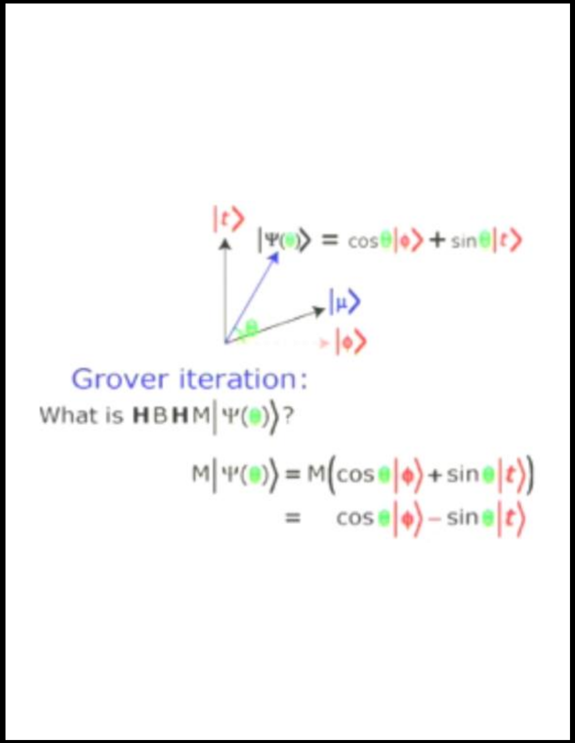
GEOMETRIC EXPLANATION

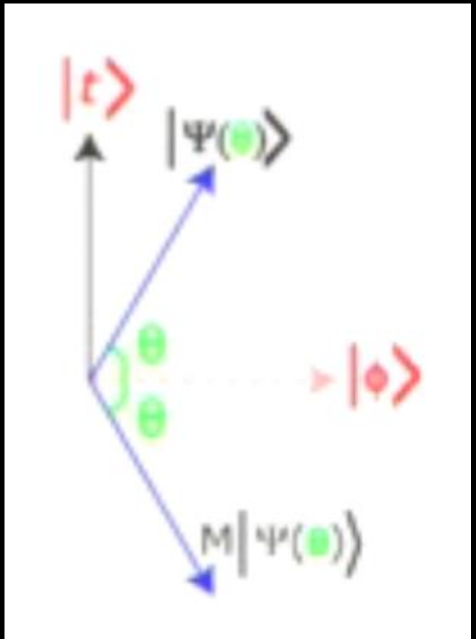
- Geometrically, all kets can be parameterized in plane through themselves and the target
- What does Grover iteration do to reach the target state?



GEOMETRIC EXPLANATION

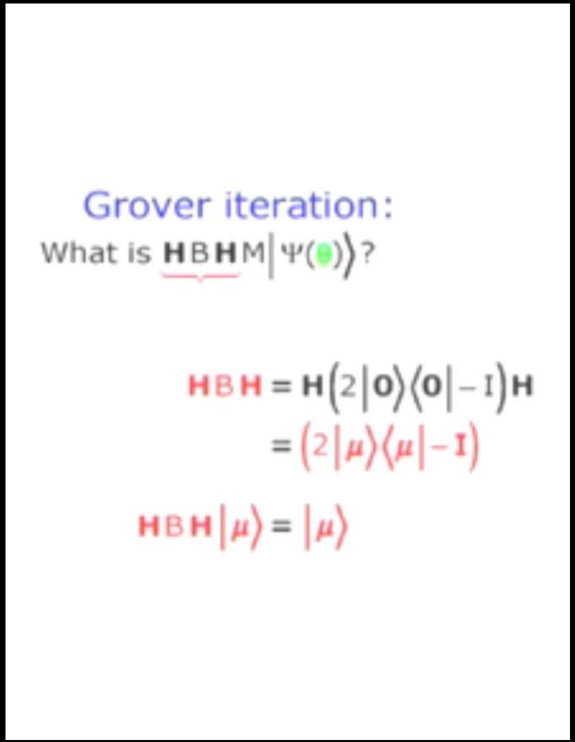
- For an arbitrary state, consider the effects of G = HBHM
- M changes the sign of the t component
- Overall effect, a reflection about the horizontal axis

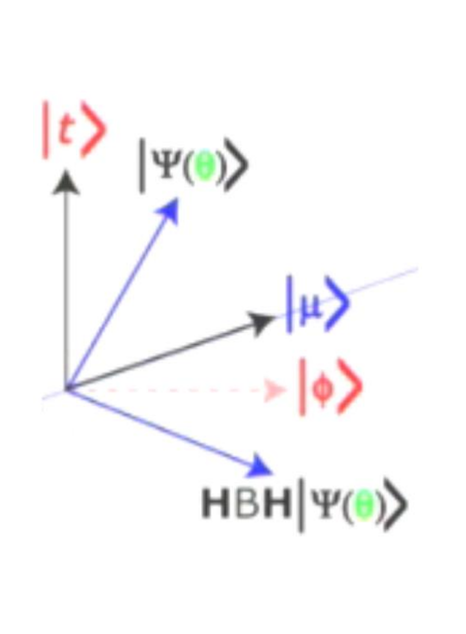




GEOMETRIC EXPLANATION

- HBH does not affect the ket |μ>
- Hence, HBH constitutes a reflection about the line through |μ>

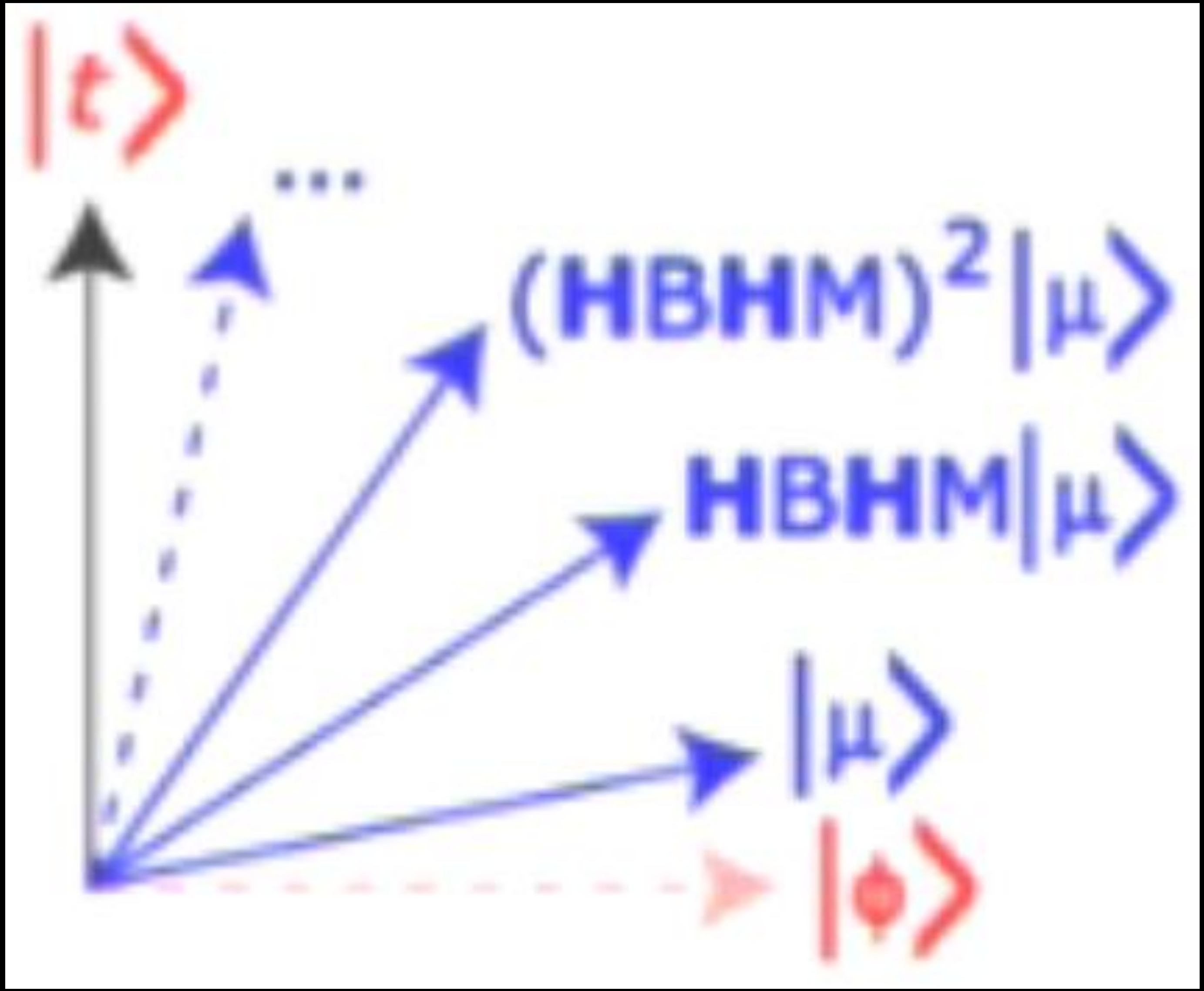




GEOMETRIC EXPLANATION

- Taken together, G produces a rotation toward the solution t
- We strategically choose our number of iterations to bring use as close to t as possible without passing

- How many iterations does this take? In fact, about $\sqrt{N}$ only, compared to N for classical algorithms



CONCLUSIONS

- Grover's Algorithm is optimal; under idealized conditions, cannot be surpassed by any quantum or classical exhaustive search algorithms
- For M desired inputs, completes search in order $\sqrt{(N/M)}$; contrasted with N/M classically
- The fast, probabilistic nature of quantum computing and search algorithms compliments the slower deterministic nature of classical computers
- Taken together, quantum computing can take problems impossible for classical computers, and make them possible to solve exactly
- We should see the realization of this potential over the next several decades