# Quantum Computing 101

( Everything you wanted to know about *quantum* computers but were afraid to ask. )

$$2^{67} - 1 = 193707721 \times 761838257287$$

- Took American Mathematician Frank Nelson Cole 20 years of Sunday afternoons to compute in 1903.
  - Wrote on board to standing ovation.
  - Famous since “Mersenne Prime”.
  - Mersenne claimed it prime in 1644.
  - Mathematica takes 0.031 seconds to do now.
- I offer $5 to the first person that can factor by hand (no calculators!) 95861 during this talk.
- Factoring can be very hard.

# Who cares about factoring?

- It is the basis of RSA, the most widely used public-key encryption system in the world.
  - Invented 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman.
  - The RSA patent expired recently.
  - RSA is a billion $ industry.


- Finding a way to factor integers quickly would allow you to break bank transactions, read secret messages all over the world, and get you a one-way trip to the National Security Agency. (And make you famous…. Or dead….)

# How RSA Works

- We encrypt a message T (as a number) into ciphertext C (another number):
  1. Find P and Q, two large (e.g., 1024-bit) prime numbers.
  2. Pick 1< E < PQ, and E and (P-1)(Q-1)=F relatively prime.
  3. Compute D, inverse of E, such that F divides (DE - 1).
  4. Encryption C = (T^E) mod PQ.
  5. Decryption T = (C^D) mod PQ.

- Public key is the pair (PQ, E). Private key is D.
- Publish public key. There are no known easy methods of calculating D, P, or Q given only public key.
- If P and Q are each 1024 bits long, the sun will burn out before the most powerful computers presently in existence can factor PQ.

# RSA Example – Easy as 1,2,3

1. I pick P=3, Q=11, E= 3, and D=7. Notice E and F=(P-1)(Q-1)=20 have no common factors, and DE=21 divided by F=20 has remainder 1. Now I publish my Public Key (PQ,E) which is (33,3). D=7 is my private key which I use to decode messages encrypted with my public key.

2. You want to send me the message 9. Use my public key and compute the remainder when 9^3=729 is divided by 33, obtaining C=3, which you send me.

3. I use my private key D (which you could too, if you could factor PQ=77)to compute the remainder of C^D=3^7=2187 when divided by 33, and I get your message back, which is 9. Amazing!

# The Point

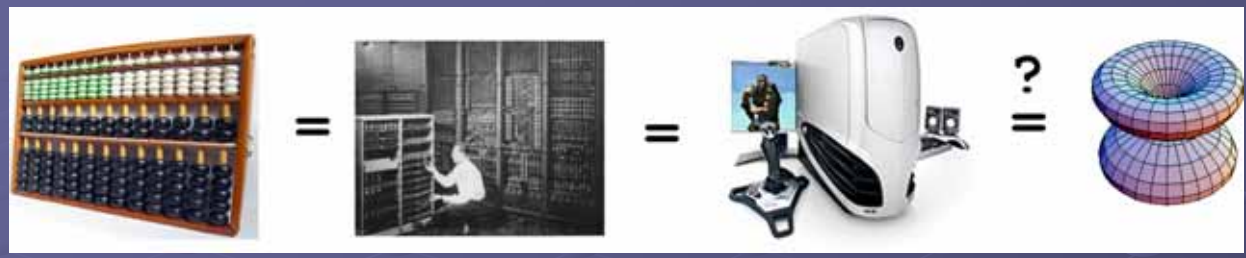
FACTORING INTEGERS IS VERY IMPORTANT AND HARD!!

$$n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$$

# Origins of Computing

- To understand how hard factoring is, we must understand what an “algorithm” is.
- David Hilbert (1862-1943)
  - Posed 23 problems in mathematics at the Second International Congress in Paris on August 8, 1900.
  - Over the past 100 years, many have been solved.
  - #10: Does there exist a universal algorithm solving “Diophantine” equations?
- Alan Turing (1912-1954)
  - Working on #10 in 1930’s, formulated what algorithm means.
  - Gives sequence for pencil and paper calculations.
  - Shows precisely what is “computable”.
  - Forms basis of modern computing.

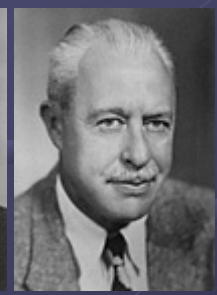
# Origins of Computing, Take Two

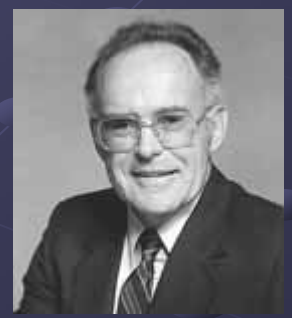



- Church-Turing Thesis: “All computers are created equal.”
  - Universal Turing Machine – can simulate any other machine efficiently.
  - Turing claimed any device which does computation is roughly as good as any other.
  - Holds until now – quantum computing seems to jump to a new level.

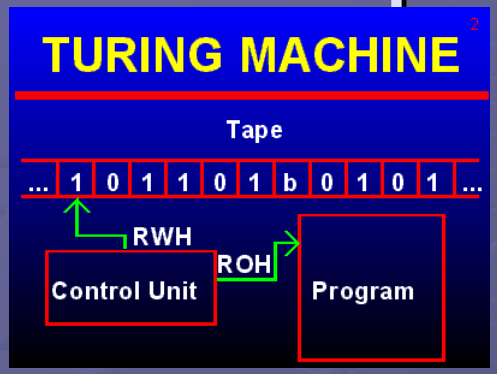
# Computing Model

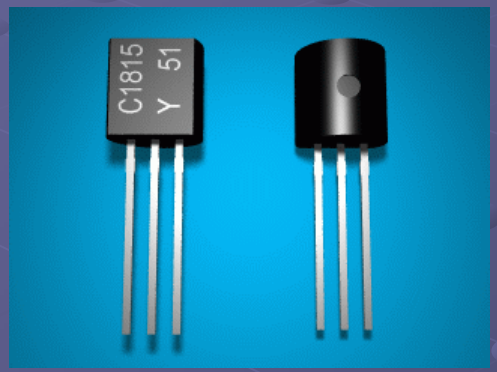
- John von Neumann developed a simple model of how to put together electronic circuits to implement a Turing machine.
- 1947 - John Bardeen, Walter Brittain, and Will Shockley invented the transistor, for which they received the 1956 Nobel Prize in Physics.
- 1965 - Gordon Moore (founder of Intel) states Moore's Law: "Computing power will double for constant cost every 18 months," which has held until now.

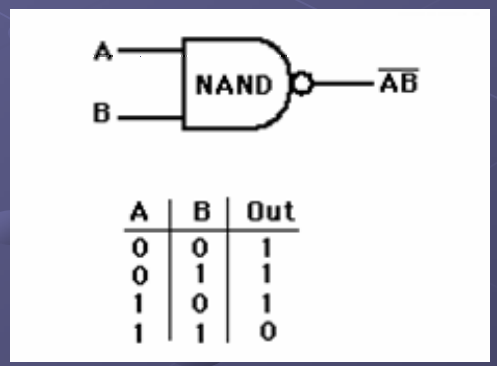




# Computing Model, Part Deux




- Turing machine
  - Mental construct to describe algorithms.
  - Idea is an infinite tape.
  - Describe how computer works: 0's and 1's idea.
  - Each cell is a "bit", stores 0 or 1.
- Basic component:
  - Physically: transistor.
  - Conceptually: NAND gate.

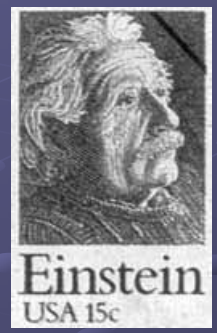
# Physical Computing Limits

- Quantum effects
  - As circuits get small, electrons “jump” out of wires and cause problems
- Some problems just too hard
  - 1982: Richard Feynman in pointed out that classical computers *cannot simulate quantum systems*, since in general describing n quantum particles require 2^n complex numbers. He suggested building quantum systems to run these simulations.
  - Example: Simulating 50 particles requires 2^50=1,125,899,906,842,624 numbers to be stored in the computer.

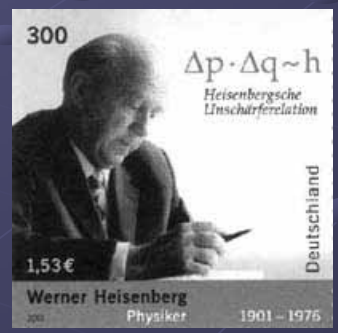
Richard Feynman 1918-1988

# The 2-Slide Quantum Mechanics Primer

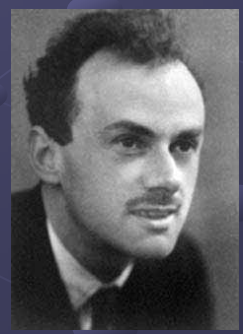
- To understand quantum computers, we must explain some quantum theory.
  - Created from about 1900 through 1940.
  - Plank (1900), Einstein(1905), Bohr(1913), Schrodinger(1926), Heisenberg(1925), Pauli(1933), Dirac(1928), von Neumann (1932).



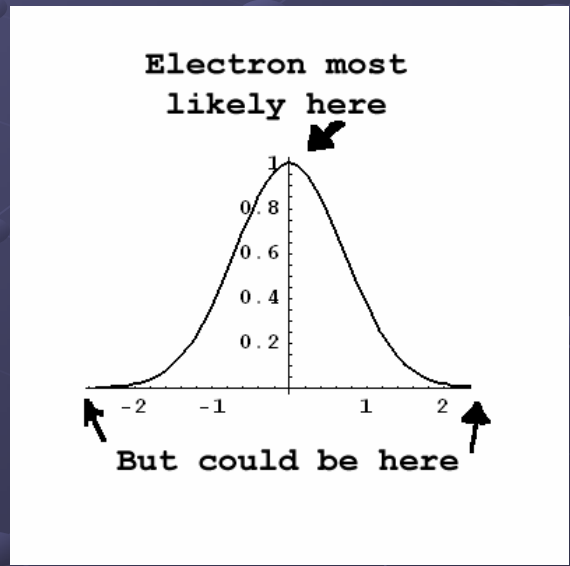




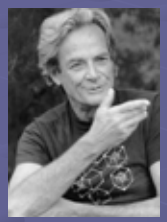
# 2-Slide Quantum Mechanics Primer

(Slide 2)

- Classical particles
  - Certainty in momentum and position
  - Observing does not change them
- Atomic particles
  - Uncertainty in momentum and position
  - Observing changes them
  - Examples: electron, proton, neutron, photon
  - Statistical in nature – “spread out”
  - Tunneling.

# Quantum Computing Origins



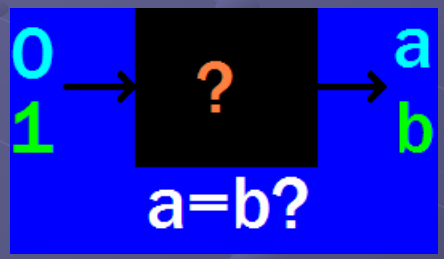
- The shrinking component size in recent chip design is running against quantum effects, so researchers were wondering how to avoid them. It turns out that they might actually help computation, if used properly.
- Feynman – labeled above noticed QM not possible on TM.
- David Deutsch
  - In 1985, trying to derive Church-Turing Thesis from physical law led to quantum systems.
  - Defined notion of Universal Quantum Computer: capable of efficiently simulating any QTM.
  - Found a problem that QTM could solve efficiently, but TM could not. Unknown if any physical system can be efficiently simulated on QTM.
- Peter Shor - 1994 - integer factoring algorithm, N digit number.
  - Exponential speedup over best known integer factoring algorithm (GNFS)
- Lov Grover – 1994 – search reduced from $O(N)$ to $O(\sqrt{N})$

# Quantum Computing Model

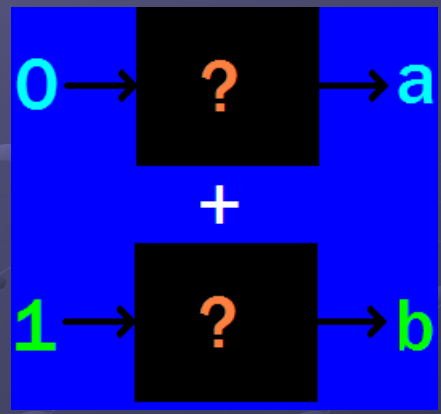
- Classically, a “bit” is the smallest piece of information a computer can store, which is labeled 0 and 1.
- Classical bits are 0 **OR** 1, never both, never somewhat of a mix….. [0] **OR** [1].
- Quantum bits (“qubits”) are allowed to be a mix of states 0 and 1, each with some probability…… [0] + [1] *at the same time*.
- Qubits can be “entangled” in a manner not possible with classical bits.

# Deutsch's Algorithm

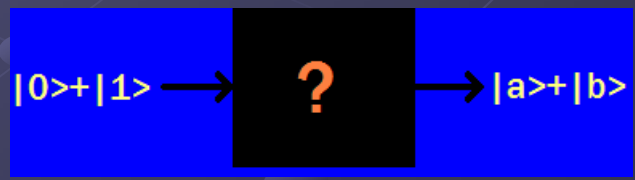
1. Problem : Given a black box that takes as input 0 or 1, and outputs 0 or 1 deterministically, decide if the box outputs the same value for different inputs.




2. Classically, this box needs to be interrogated twice.

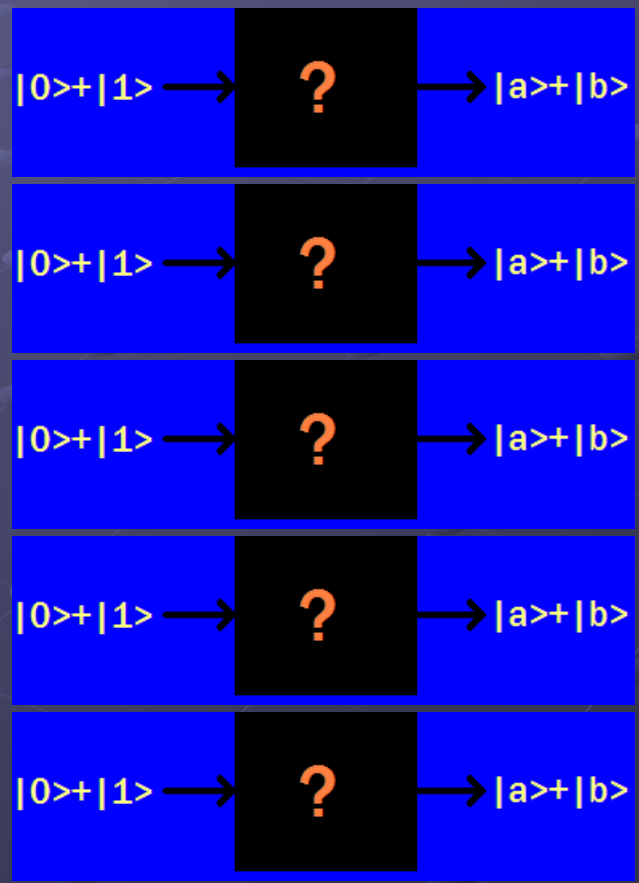



3. If it is a quantum box, handling superpositions of states, it only needs to be interrogated once.



# Simon's Algorithm

- Generalized Deutsch's algorithm to many questions at once.
- Showed a problem that *requires* exponential time on any Turing machine, but is polynomial time on a quantum Turing machine.
- Result: Quantum Computers are **MORE** powerful than any classical computer can be.

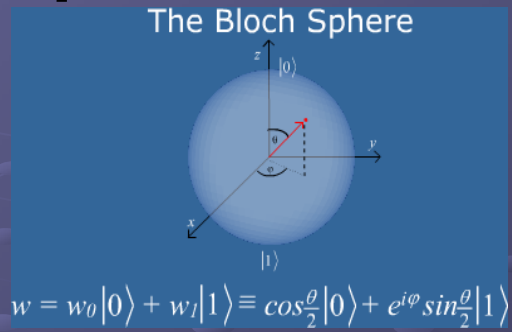


# Shor's Factoring Algorithm

$$\exp\left( c(N)^{\frac{1}{3}} (\log N)^{\frac{2}{3}} \right) \rightarrow N^2 (\log N)(\log\log N)$$

- Obtained from studying Deutsch's and Simon's algorithms.
- Reduces factoring to *period finding.*
- Exponentially faster than best known classical algorithm.
- For example, it would require 1000 workstations $10^{29}$ years to factor a 4096 bit number using GNFS, but take a 100 MHZ quantum computer only 4.8 hours.
- Algorithm was experimentally verified in 2001 by IBM, using 7 qubits to factor 15 = 3 x 5.

# Factoring Comparison




| Year vs bitsize | 1024 bits | 2048 bits | 4096 bits |
|---|---|---|---|
| 2006 | $10^5$ | $5x10^{15}$ | $3x10^{29}$ |
| 2024 | 38 | $10^{12}$ | $7x10^{25}$ |
| 2042 | 3 days | $3x10^8$ | $2x10^{22}$ |

***Years*** required, assuming Moore's Law, no new theorems, GNFS, and 1000 workstations

| Bits | 1024 | 2048 | 4096 |
|---|---|---|---|
| Qubits | 5,124 | 10,244 | 20,484 |
| Gates | $3x10^9$ | $2x10^{11}$ | $2x10^{12}$ |
| Time | 4.5 | 36 | 288 |

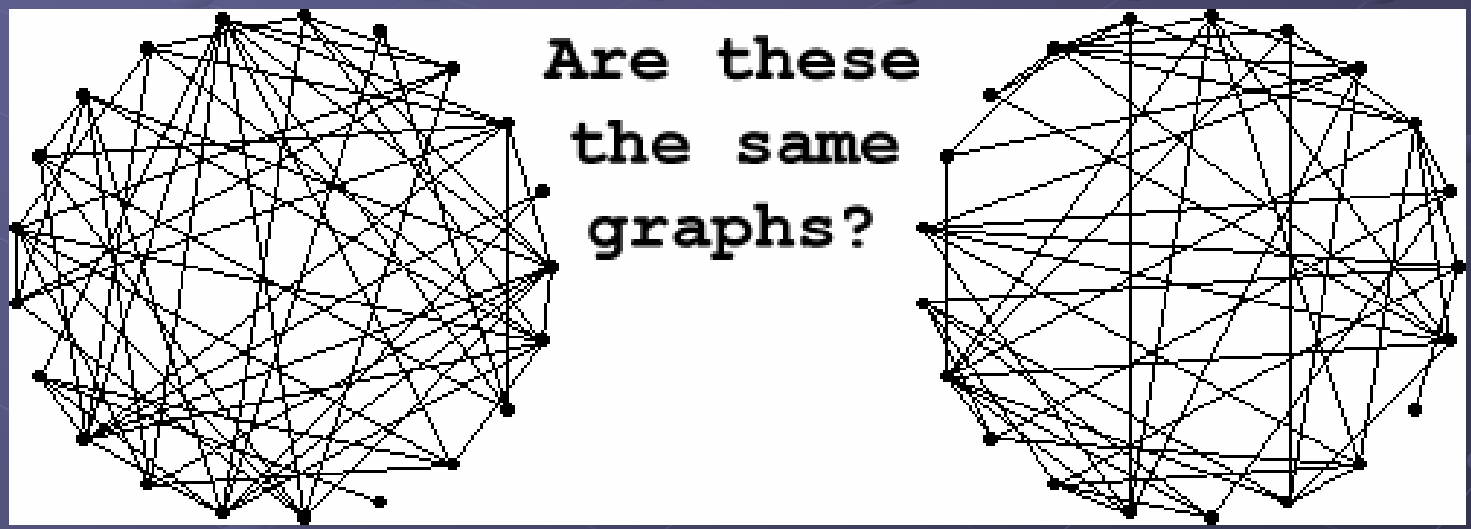
***Minutes*** required and physical requirements, assuming a 100 MHZ QC

# Grover's Algorithm

- Finds items in an unsorted list faster than any classical computer.
- Instead of using N steps, only uses Sqrt[N], by using quantum entanglement to search many places at once.
- Example: Find one item from a million
  - Expect to look at 500,000 items (classical).
  - Expect to look 1,000 items (quantum).

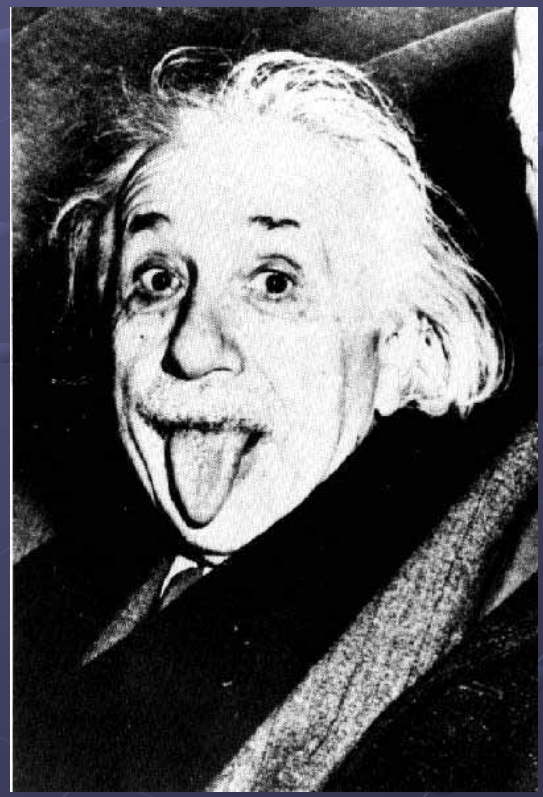
# The Grand Challenge

- Graph Isomorphism
  - Current research seeks a quantum algorithm that would give an efficient graph isomorphism algorithm.



# Future Directions

- P=NP?
  - Solution worth over $1,000,000.
- String computing?
  - Does adding relativity increase power?
- Engineering problems
- Error correction



“The most beautiful thing we can experience is the mysterious.”