D-Wave One

- May 11, 2011. D-Wave announced the D-Wave One, a 128-qubit quantum computer, that they claim is the world's first commercially-available quantum computer. In fact, someone's already bought one: # Lockheed Martin.
    - Unsurprisingly, the # internet # news sites # went # absolutely # crazy.
    - Also unsurprisingly, people on the internets went crazy. # # # # This is all a bit, well.... #
    - The biggest internet claim: THIS MEANS ENCRYPTION IS # doomed, # DOOMED, # DOOOOOOMED.
    - Our financial system is # GOING to BLOW UP.
    - So WTF is up? Why does a quantum computer mean we're, you know, # screwed? And is D-Wave One the real deal?
- # [TITLE] So let's talk about what D-Wave One is -- a QC. What's the promise of quantum computing? What is D-Wave really selling?
    - There are a lot of misperceptions about what QC is. # Anyone here know what a QC is? (if so: Excellent! If you'd give my talk for me then I could catch up on some sleep)
    - To explain why people are interested in quantum computing, let me talk about normal computing and then explain how a quantum computer is different
    - Caveat: I'm going to be telling you analogies and doing some hand-waving, so I'll be lying. But that's okay! If this gets you excited, then I can point you to the research to learn more.
- Right, so normal computing. # Classical computers work with bits.
    - [Four people @ front of room w/two flashcards. 0 or 1 on front, 0 or 1/2/4/8 on the back. Arrange them in order. Get audience to select which ones are 0s and which ones are 1s. From that, make a decimal number.]
    - # Here's a binary number. # It's base 2. Each of those 1s or 0s stand for a power of 2. # Here's what the binary number is in decimal.
- Computers shove bits around. There's a limit to how fast you can do that, though. How do you speed up computers?
    - # Maybe "Moore's law" will save us. Gordon Moore, co-founder of Intel, famously said that the number of transistors on a computer chip doubles every two years, which means computers are getting exponentially faster. # Here's a list of Intel processors from 1970 on, and the number of transistors they have. # Look! It's a straight line, which on a log plot like this means it's exponential. Moore's law has been true for four decades!
    - That won't go forever, though. Right now we do transistors through photolithography, and increasing its resolution gets harder and harder. Eventually you'll have molecule-sized transistors, at which point quantum effects start to play hob with what you want to do. Like, the transistors are so small that electrons going down one trace and into one transistor might tunnel over to a neighboring transistor. These are quantum effects showing up unexpectedly and messing things up, and not quantum computing.
    - Okay, so add more processors! It's like all of those 2, 4 and 8 processor chips. But # doubling the number of processors roughly doubles the number of size. You can do more computations in parallel, but it takes more space and power. In physics speak, a classical computer's parallelism goes up linearly with system size.

- # Richard Feynman talked about this problem back in the 1980s. There are quantum mechanical effects that you just can't simulate on a classical computer: it takes too much resources, even now, nearly 30 years later. (And we'd really like to be able to simulate quantum mechanics for research purposes, and we already have honkin' huge classical computers.) But nature does it all the time! So could you do computing using quantum effects?
- Let's talk about quantum effects and how we can compute with them.
- I'm going to explain quantum effects to start with using light polarization. # Light can be made to wiggle in one direction. That's polarization.
    - [2 volunteers to do laser pointer and polarizers. 0, 90, and 45 degree polarizers. Given unpolarized light, polarizers let half the light thru. WTF? Polarizers can't just be knocking out all photons that aren't going in the right direction, or you'd get a lot less than ½ the light thru. Also: 45 degree polarizer lets light thru. WTF???]
    - Polarization is a quantum effect. In quantum mechanics we talk about "quantum states" and write them with # these brackets. (Don't worry about what exactly the weird brackets neab right now. It's a math notation called Dirac notation. I'm using it here as a flag to say, "Hey, this is quantum!")
    - You can be in a mix of pure states. Here, up/down and left/right are the pure states. When I throw in a polarizer at 45 degrees, # I get light that's a mix of those two states. The number in front of the states are the amplitudes.
    - (Amplitudes can be negative, positive, or complex.) # If you square them, you get probability of being in that state.
    - So this mix of states is called a "superposition". When you go to measure the state, though, you collapse the waveform. The superposition of states becomes a single state. In this case up here, you've got a 50% chance of measuring it in up-down, and 50% chance of measuring it in left-right.
    - The polarizer is measuring the state. # The vertical polarizer is making a measurement and collapsing the waveform so all of the light is now vertical. Then the horizontal polarizer measures nothing.
    - # Put a 45 degree polarizer in there, though, and boom! You get a superposition. The horizontal polarizer then collapses the waveform and you've got a 50/50 chance of any photon being an up photon. That's why you lose ½ the light at that step.
    - Take-home message: 1. We talk about quantum states. 2. System can be in more than one state at a time, like how up-and-right polarized can be superposition of up and right.
- What happens if you build a computer bit out of quantum states? Well, you get a # qubit. No, *qubit*. # Okay, whatever. You can make a qubit out of any particle/system with two possible states. One state will be 0, and the other will be 1. Like, for light, up/down light could be 0 and left/right could be 1.
- So what does that get us? # Consider 3 bits. They can hold any number from 0 to 7. Now consider 3 qubits. They can hold any number from 0 to 7 *at the same time*. (P.S. | 000> is just a compact way of writing all three qubits' individual states.)
- But what good does that do us? So a qubit can hold multiple numbers at once. Big deal.
    - Actually, it is. (Get a bag or hat from the audience.) Imagine adding two numbers and getting a third. (Put cards w/the addition on them in the hat.) Classically, you add one number to another and you get a third. But imagine you're working with qubits and both numbers are in a superposition of states -- they have "more than one number" in them. # If you add the two qubits together,

you're adding all possible combinations of the numbers together!

- Because of this superposition, the amount of parallelism goes up exponentially w/size. (Remember: in classical computers the parallelism goes linearly with size. Double the size, double the parallelism.)

- Drawback: you can do all of these parallel computations. But when you go to look at the result, you collapse the waveform! You only get one answer out! (Draw answer from hat.) Even worse, the answer you get is random! #
    - There are ways around this, but you have to be tricksy. ("manipulate the quantum state so that a common property of all the output values such as the symmetry or period of a function can be read off" as in Shor's algorithm, or "transform[s] the quantum state to increase the likelihood that the output of interest will be read" as in Grover's search algorithm) So it's not quite right to say that quantum computers can solve hard problems in a single shot -- they may calculate all possibilities, but you can't get the right one unless there's e.g. an underlying structure you can exploit so that nearly all measurements get you the right answer.
- So what kinds of problems can you solve with quantum computers? Big one is Shor's algorithm. It factors big numbers.
- Modern # public key cryptography depends on how hard it is to factor big numbers.
    - Hand-waving time. Pick two large prime numbers at random, which we'll call p and q. You multiply them together to get p*q=n. Roughly speaking, n is part of your public key, which you make public. p and q are part of your private key, which you keep secret. You can encrypt a message using the public key, and only the person who has the corresponding private key can decrypt it. This is how the RSA algorithm works.
    - Here's the thing: if you could factor n back into p and q, you'd have the private key. But factoring integers -- figuring out what numbers you can multiply together to get the original number -- is hard work for a computer. If p and q are very very very large, then n is even larger, and computers take a long time to figure out the factors. If n is 256 bits, you can break it using my laptop here in a matter of hours. For 520 bits, give me a couple of PCs and a couple of months and I'll break it. If n is 1024 bits, you could build a piece of so-far hypothetical hardware for, oh, around $10M to $50M that would break the key in a year (TWIRL). If n is 2048 bits, well, you'll be safe for quite a while.
- A quantum computer could try all possible combinations of numbers all at once in parallel. # Genius! Except when you went to measure the answer, you'd get one of all the possible divisors at random, and it'll probably be ones that don't work. # Oops. You're looking for a # needle in a haystack and making random grabs into that haystack hoping you stick your finger. You might as well use a classical computer.
- Fortunately, there's a clever work-around. It turns out that # there's a way to find the factors that involves finding the period of a sequence. A sequence of numbers that repeats itself has exactly one period -- a period is how long the sequence goes before it repeats itself.
    - (Get six men and two women. Arrange them M M W M M W M M. What's the period? It's 3! Every three people we start back the pattern of men vs. women. Arrange them M M M W M M M W. The period is 4!)
- Now we're getting somewhere. We're exploiting a hidden structure of the problem: that factorizing numbers can instead be turned into looking for the period of a sequence of numbers by making the sequence (x mod n, x^2 mod n, x^3 mod n, x^4 mod n, ...), and if you find the period of that sequence, you've found the divisors. (I'm glossing over stuff, but hey, the Internets can tell you more later.)
- Now, you can do this algorithm on a classical computer -- and in fact most classical factorization algorithms do! But the sequence of numbers you have to look at before you start to repeat can be nearly as large as n. That's a lot of numbers. So doing

factorization this way doesn't make life easier.

    - Except! We have a quantum computer. Mua ha ha ha! And there's a tricky way of making a superposition that has all of the sequence of numbers stored in it, and a way through a quantum Fourier transform that you can turn each and every state in that superposition to be the period of the sequence of numbers. Now, when we look, no matter what state we end up reading out, we get the right answer. We've turned our # needle in a haystack into a # giant pile of needles.
- # In 1994 Peter Shor created the above algorithm. A regular computer figures primes in $O(2^{(\log n)^{1/3}})$. Shor's algorithm works in $O((\log n)^3)$. In non-mathy terms, it works very very fast.
- Okay, so can we do this with D-Wave One? # Destroy banking as we know it???
    - Actually, # no. D-Wave One isn't built to run something like Shor's algorithm through digital computation. It's doing something very different: quantum annealing. It's like analog quantum computing instead of digital quantum computing with logic gates and whatnot, and it's not doing universal adiabatic quantum computation.
- In computing, annealing is a way of solving hard optimization problems. For instance, the # Traveling Salesperson problem. See, long before the internet, there were these people called "salespeople" who would travel from city to city hawking their wares. Let's say you've got a salesperson who needs to drive to a bunch of cities and wants to visit them in an # order that minimizes how far he has to drive. This is one of those problems that, if you give me a solution, I can check it really easily. However, to come up with the solution in the first place is hard.
    - The answer: simulated annealing. Annealing is a physical process where you # heat up a metal so that, when it cools, its atoms realign itself and you relieve internal stress. What's happening is that the atoms are settling into their lowest-energy state. The heat lets them get unstuck from a local minimum.
    - # Simulated annealing does something like it. Pick a starting solution. Have a fitness function that tells you how good the solution is. (For Travelling Salesperson it's the length of the drive.) Now choose a random nearby solution, like swap the order of two cities in the drive. If the # new solution is better than the old one, or at least only a little worse, then use that new solution. In general that means that you'll move "downhill".
    - At first "nearby" isn't all that nearby -- sort of like when the temperature is hot in the metal. But as the simulation goes on you make "nearby" be a smaller and smaller change. # This helps keep you from being stuck in what's called a local minimum.
- Okay, but how does quantum make it better? If you're stuck in a local minimum, you have to "thermally" jump over barriers. # Quantum mechanically you can tunnel through those barriers. In addition, you can in practice try several moves at once through the parallelism inherent in qubits.
    - Here's a secret: we still don't know how much this'll speed up computation in practice. There are theoretical best cases where quantum annealing is faster than conventional processing. There are theoretical worst cases where it's just as slow. We won't know how well it "typically" behaves until people have time to play with D-Wave One and try it out.
- So what do you do with D-Wave One?
    - Binary classification: a supervised machine learning technique in which you feed a data set in and tag each element in the data set with one of two tags. So, for instance, # "Does this picture have a car in it Y/N" or "Will you go out with me Y/

N".

- Turns out there's money in this kind of binary classification. Imagine being able to sort through parts on an assembly line and throw away the bad ones. (Computer chips, anyone?)
- You can also use this to train non-quantum decision-making systems. Systems like # neural networks have to be trained -- you have to develop a series of weights for the network. D-Wave One can, in theory at least, not only create the weights but also develop the neural network at the same time.

D-Wave One
Stephen Granade
Dragon*Con 2011

- What does D-Wave One look like?
    - # Regular computer -> Ranier processor (16 cells with 8 qubits each) -> regular computer. You program it using an API that supports Python, Matlab, C++, Java.
- Great! I want one!
    - It'll only cost you # $10M. And that's before the ongoing costs -- staff to keep it running, staff to program it, the fact that it's a computer in a # giant liquid helium fridge that's inside a Faraday cage inside a giant room.
    - As I said early on, they've sold one so far, to Lockheed Martin, along with all the support people. LM are going to play with it, though they haven't said what for. Who knows what they'll do with it. #
- # SUMMARY SLIDE So D-Wave One isn't going to be able to factor large numbers any time soon, but it may introduce a new way of doing classification and decision-making. In fact, if you're expecting quantum computers to take down our financial system, you may be waiting a while. Researchers *have* run Shor's algorithm -- to factor the large prime number "15". Researchers need to make the tools that will eventually lead to something scalable, where you can make a quantum computer with a lot more qubits using the same tech, where you can initialize the qubits like you need them, it runs fast enough that decoherence isn't a problem (or you have good enough error correction), and you can read out the qubits easily. To me, that's exciting. There's a huge amount of nifty science to be done!